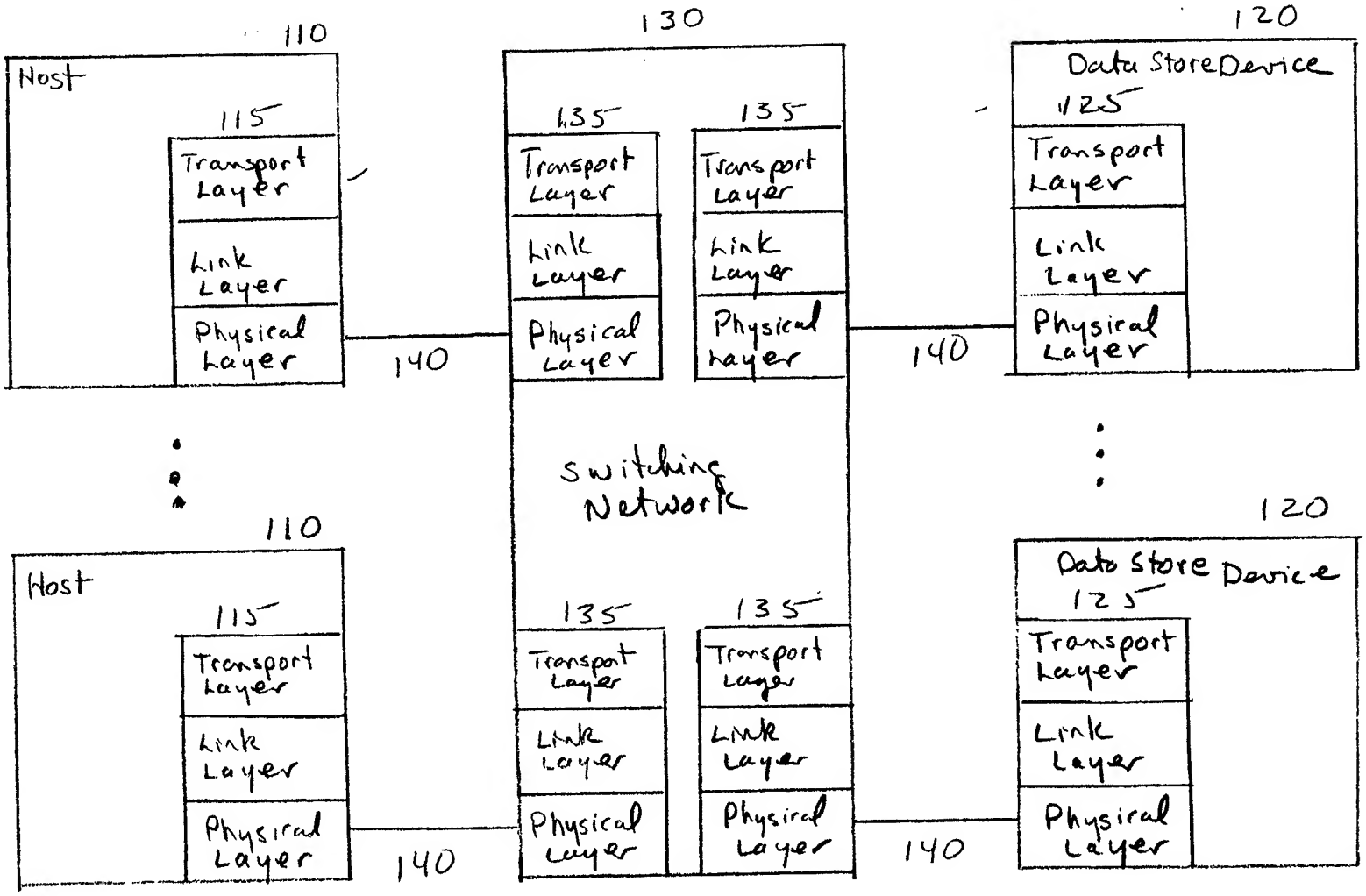


Fig 1

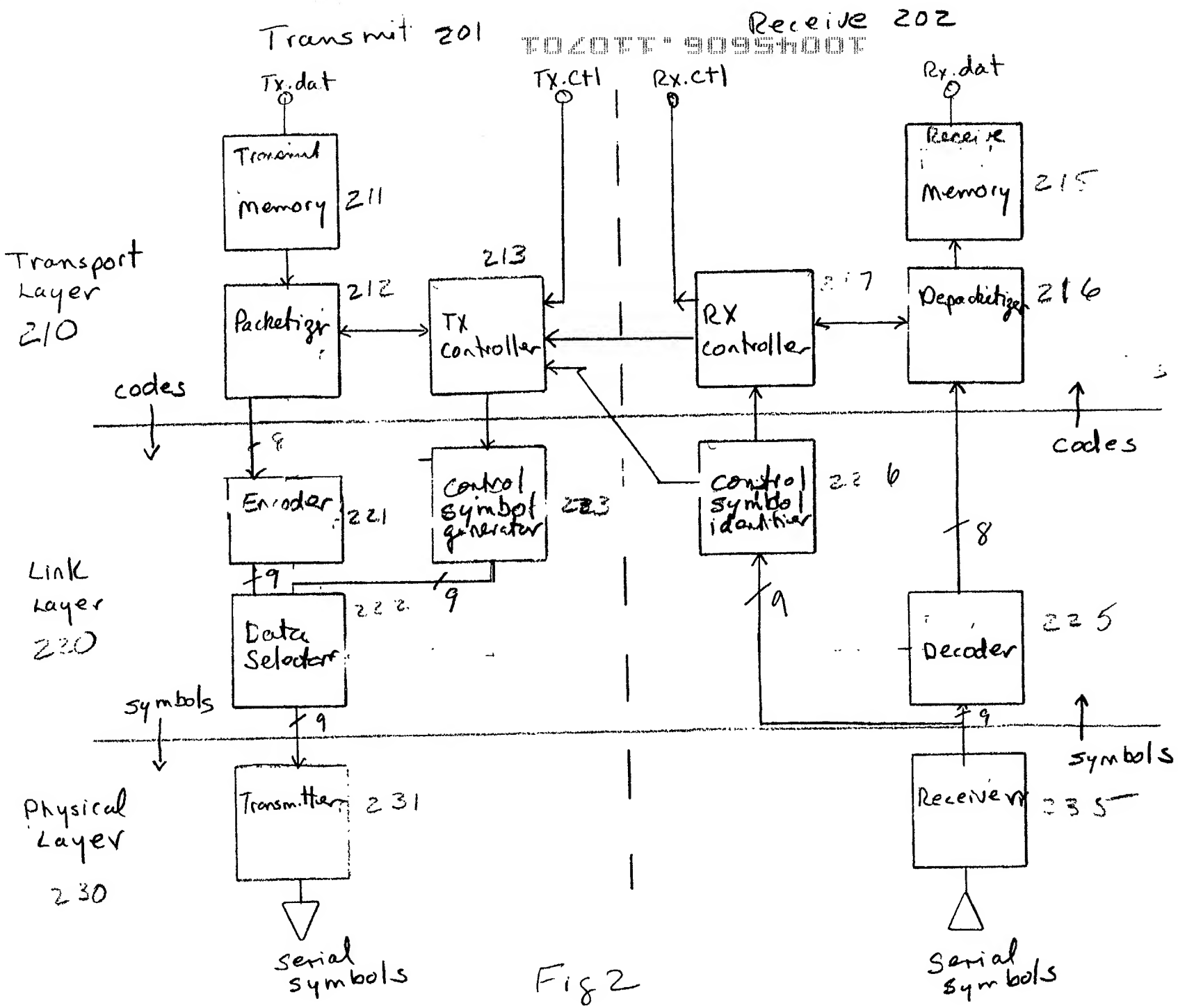

Transmit 201
Receive 202
Tx.dat
Tx.Ctl
Rx.Ctl
Rx.dat
Transmit memory 211
Packetizer 212
TX controller 213
RX controller 217
Receive memory 215
Depacketizer 216
Transport Layer 210
codes
codes
8
Encoder 221
Control symbol generator 223
Control symbol identifier 226
8
9
9
9
Link Layer 220
Data Selector 222
Decoder 225
symbols
9
9
symbols
Physical Layer 230
Transmitter 231
Receiver 235
serial symbols
Serial symbols

Fig 2

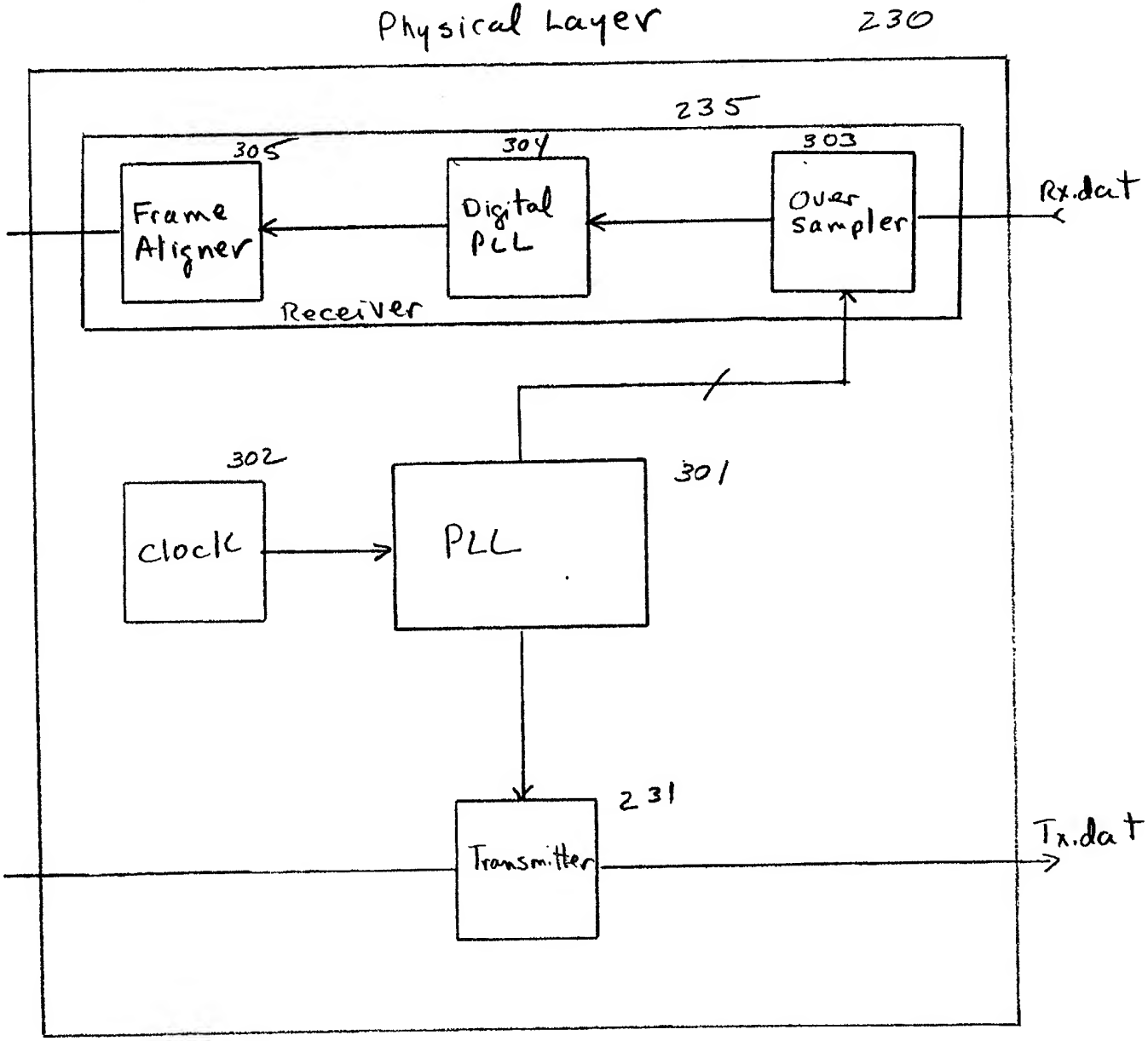

Physical Layer
230
235
305
Frame Aligner
304
Digital PLL
303
Over sampler
Rx.dat
Receiver
302
clock
PLL
301
231
Transmitter
Tx.dat

Fig 3

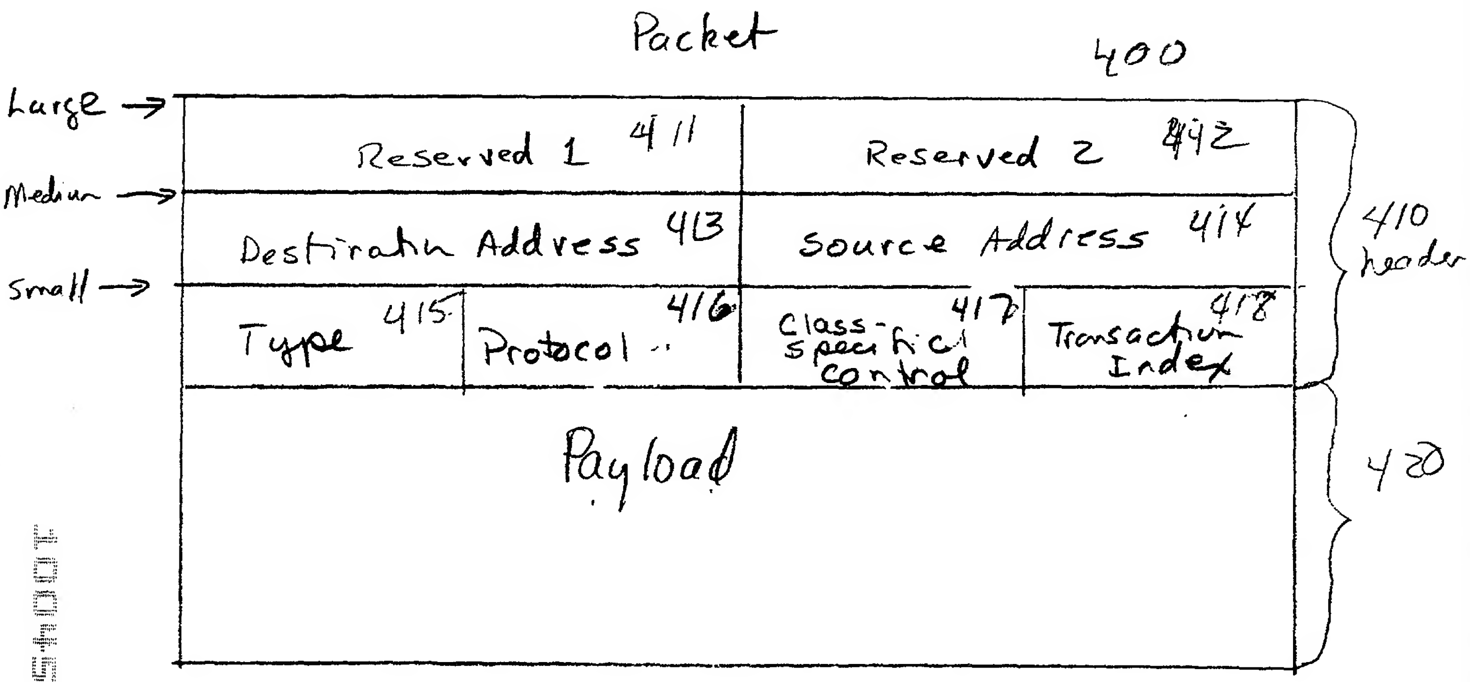
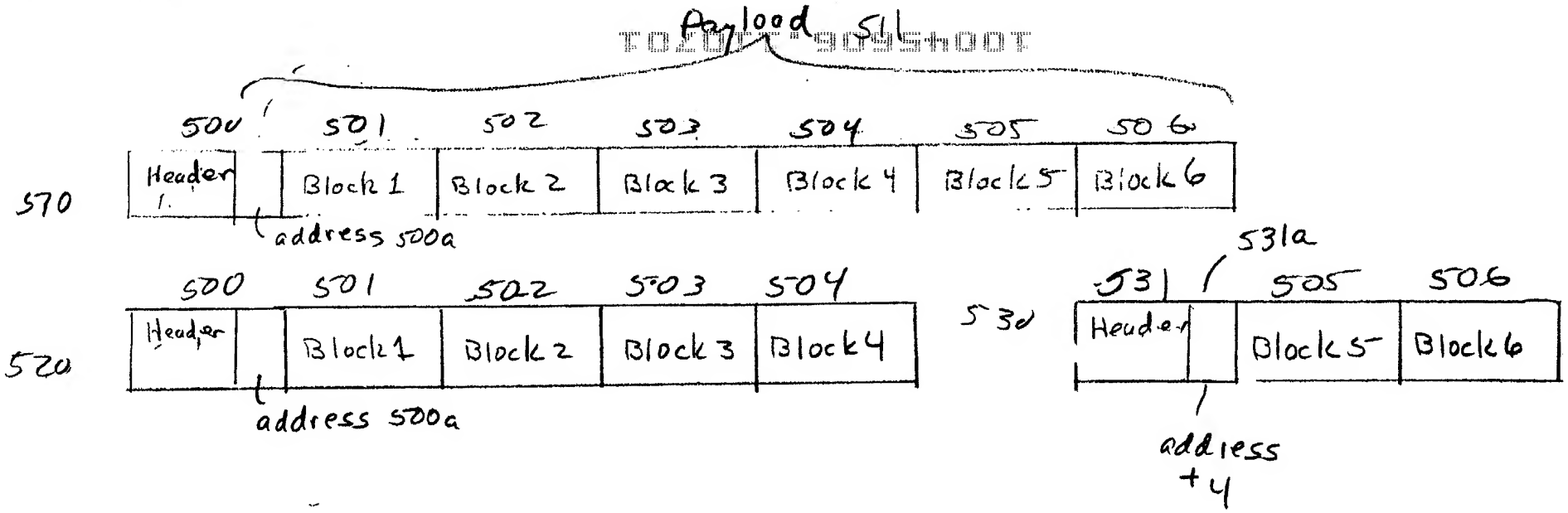

Packet
400
Large
Medium
Small
Reserved 1 411
Reserved 2 412
Destination Address 413
Source Address 414
Type 415
Protocol 416
Class-specific control 417
Transaction Index 418
410 header
Payload
420

Fig 4


Payload 511
500
501
502
503
504
505
506
570
Header
Block 1
Block 2
Block 3
Block 4
Block 5
Block 6
address 500a
500
501
502
503
504
520
Header
Block 1
Block 2
Block 3
Block 4
address 500a
531a
531
505
506
530
Header
Block 5
Block 6
address
+4

F.g 5

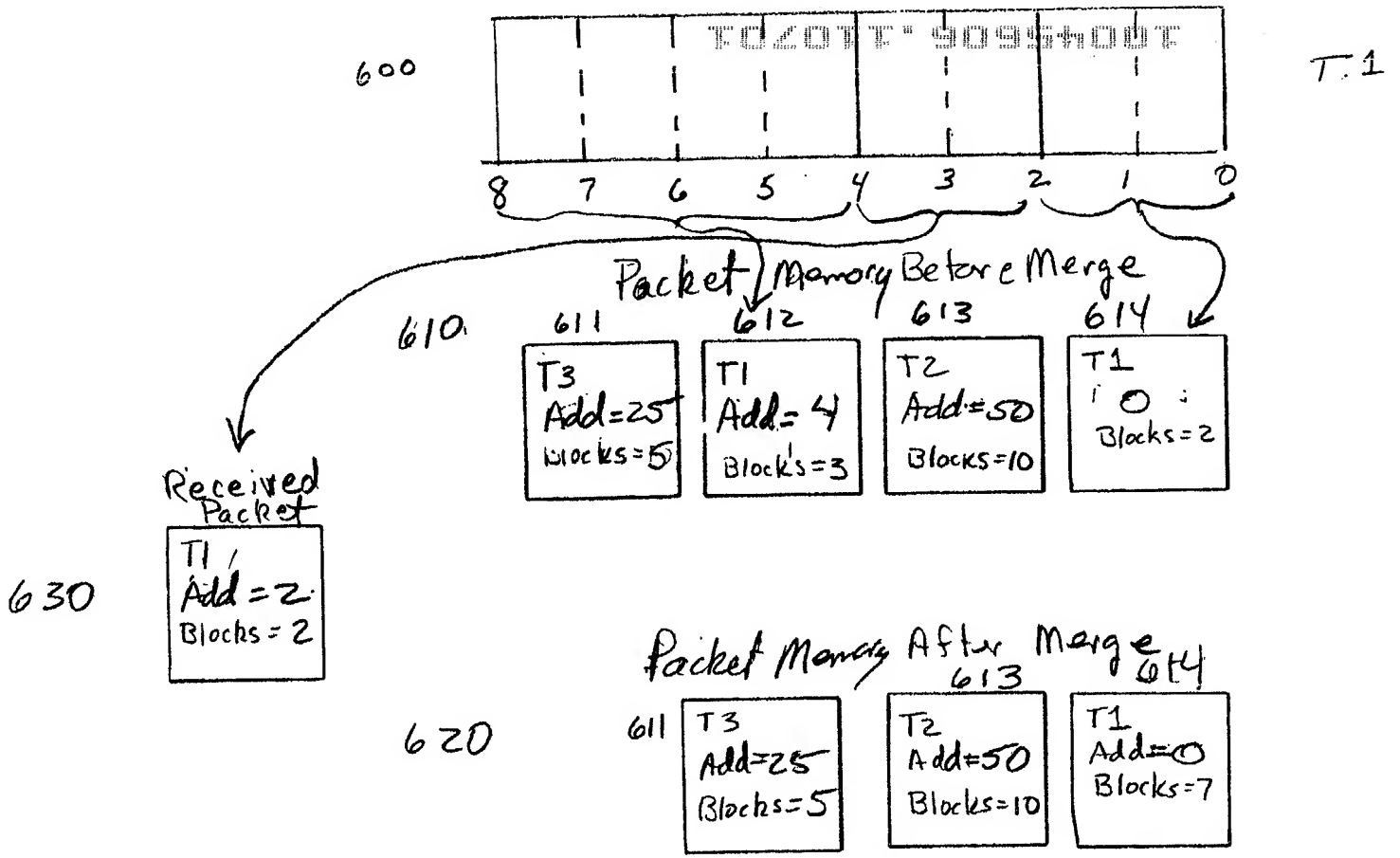

600
T.1
8 7 6 5 4 3 2 1 0
Packet Memory Before Merge
610
611
612
613
614
T3
Add=25
Blocks=5
T1
Add=4
Blocks=3
T2
Add=50
Blocks=10
T1
Add=0
Blocks=2
Received Packet
630
T1
Add=2
Blocks=2
Packet Memory After Merge
620
611
613
614
T3
Add=25
Blocks=5
T2
Add=50
Blocks=10
T1
Add=0
Blocks=7

Fig 6

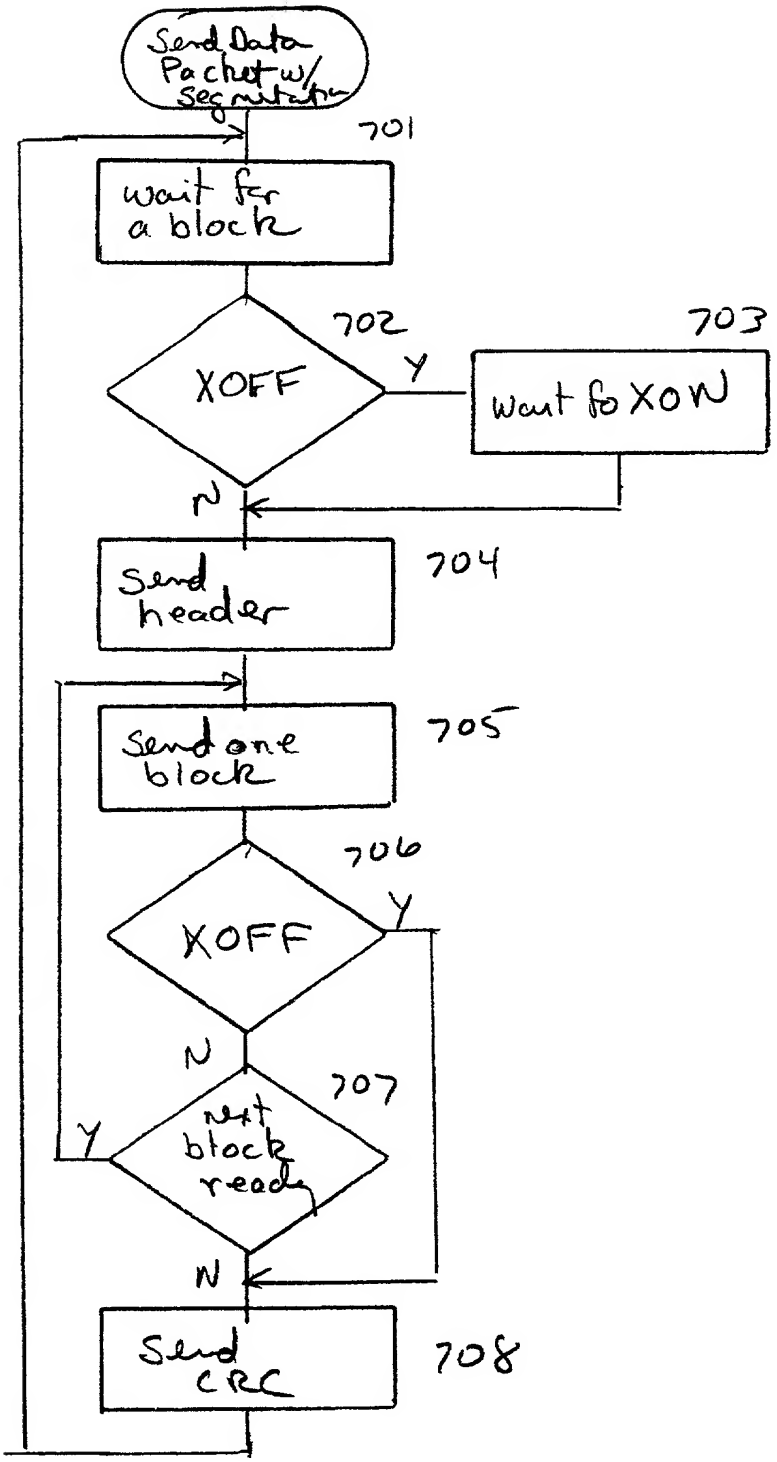

Send Data Packet w/ segmentation
701
wait for a block
702
XOFF
Y
703
Wait for XON
N
Send header
704
Send one block
705
706
XOFF
Y
N
707
Next block ready
Y
N
Send CRC
708

Fig 7

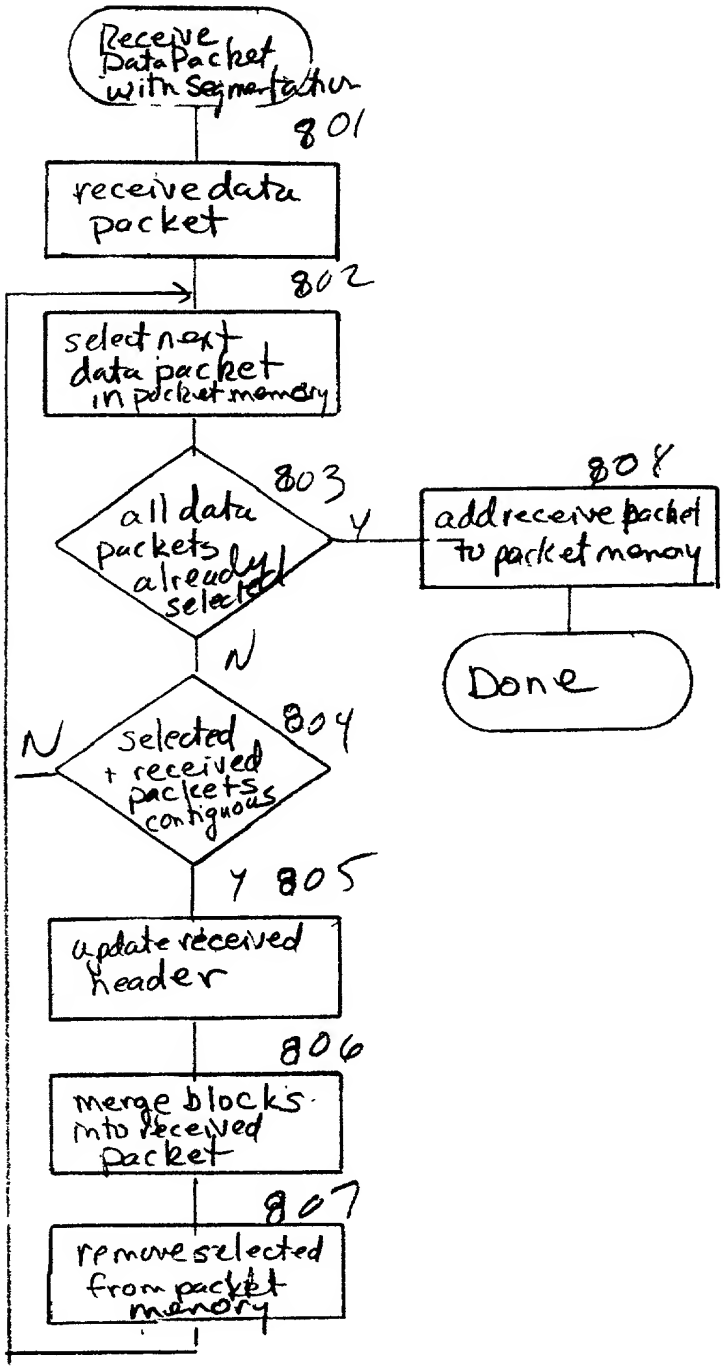
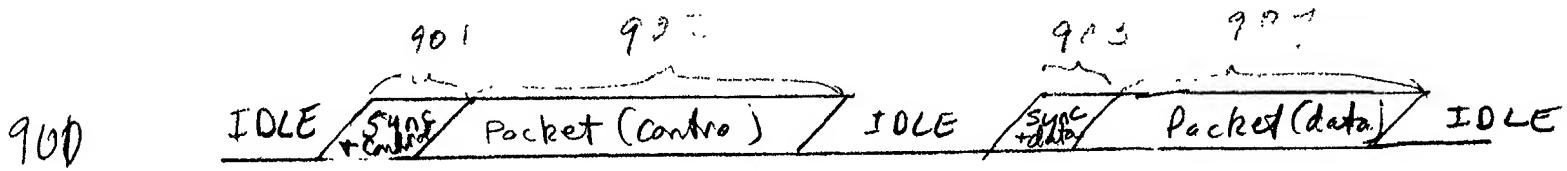

Receive Data Packet with Segmentation
801
receive data packet
802
select next data packet in packet memory
803
all data packets already selected
Y
808
add receive packet to packet memory
Done
N
804
N
selected + received packets contiguous
Y 805
update received header
806
merge blocks into received packet
807
remove selected from packet memory

Fig. 8



Sync + packet type

Fig 9A

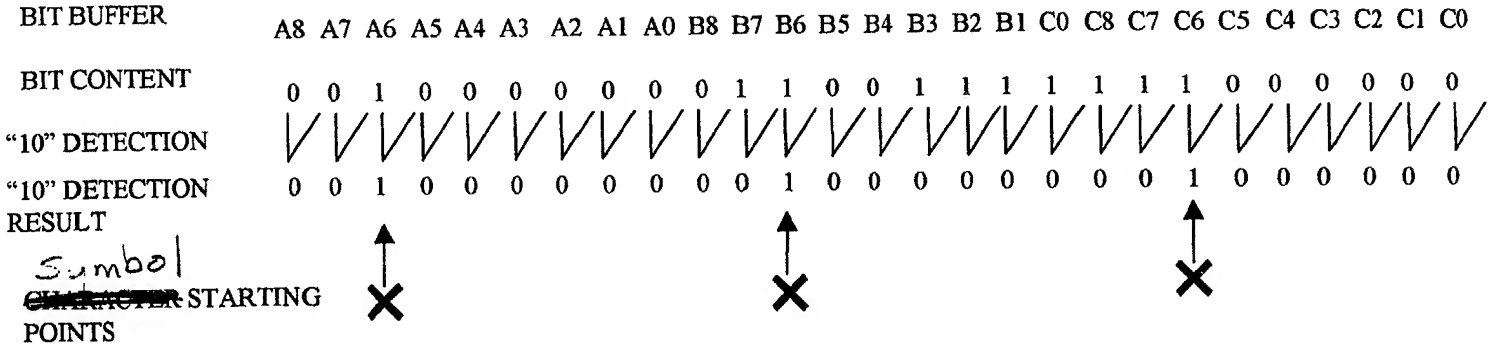

BIT BUFFER
A8 A7 A6 A5 A4 A3 A2 A1 A0 B8 B7 B6 B5 B4 B3 B2 B1 C0 C8 C7 C6 C5 C4 C3 C2 C1 C0
BIT CONTENT
0 0 1 0 0 0 0 0 0 0 1 1 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0
"10" DETECTION
"10" DETECTION RESULT
0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
Symbol
STARTING POINTS

FIG.10

Fig 9B

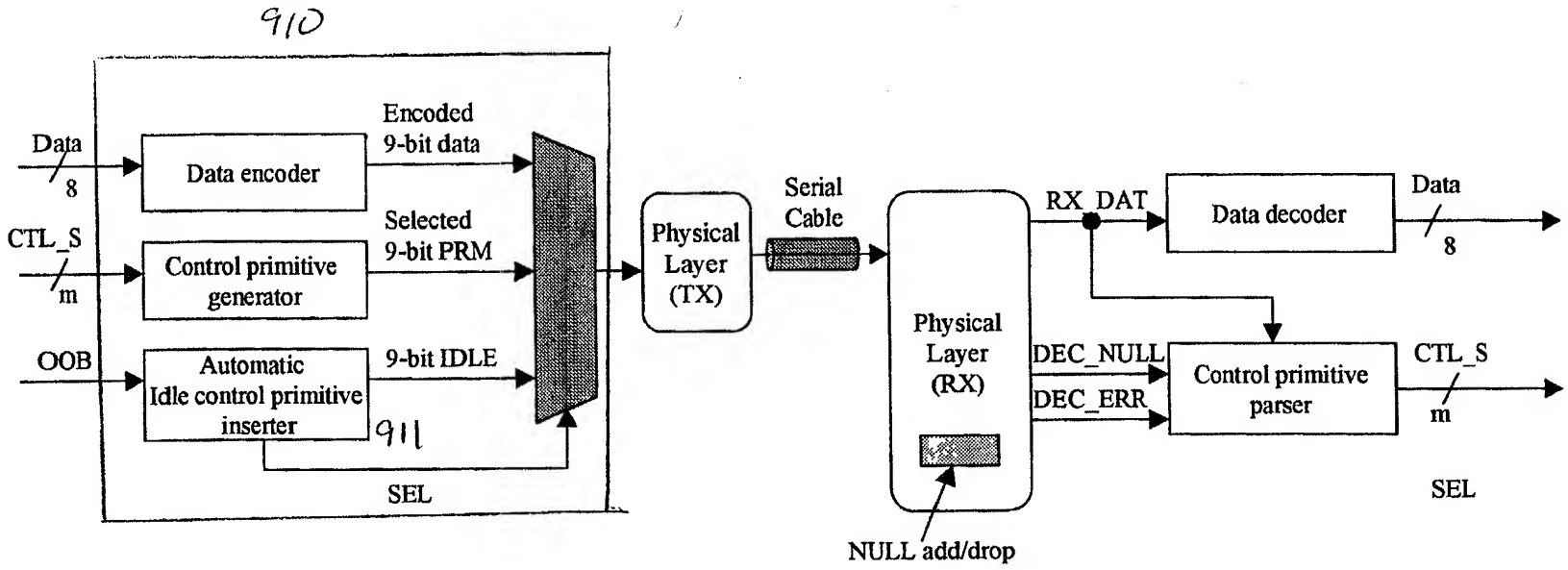

910
Data
8
CTL_S
m
OOB
Data encoder
Control primitive generator
Automatic Idle control primitive inserter
Encoded 9-bit data
Selected 9-bit PRM
9-bit IDLE
911
SEL
Physical Layer (TX)
Serial Cable
Physical Layer (RX)
NULL add/drop
RX_DAT
Data decoder
Data
8
DEC_NULL
DEC_ERR
Control primitive parser
CTL_S
m
SEL

Fig. 9C

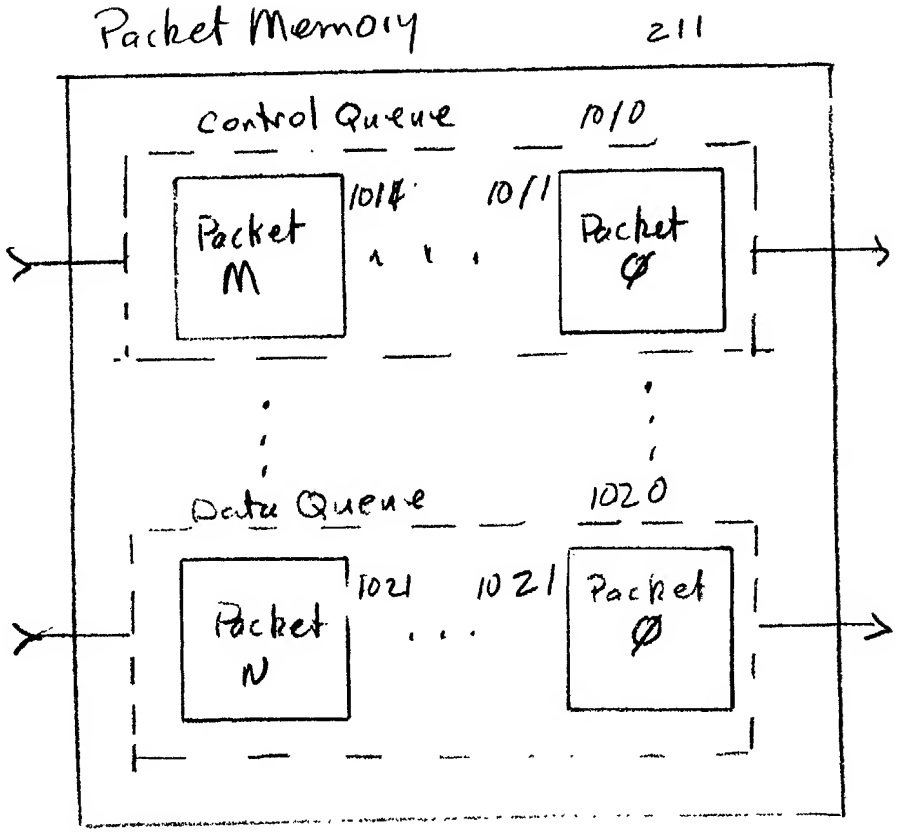

Packet Memory
211
Control Queue
1010
1014
Packet M
1011
Packet Ø
Data Queue
1020
1024
Packet N
1021
Packet Ø

Fig 10

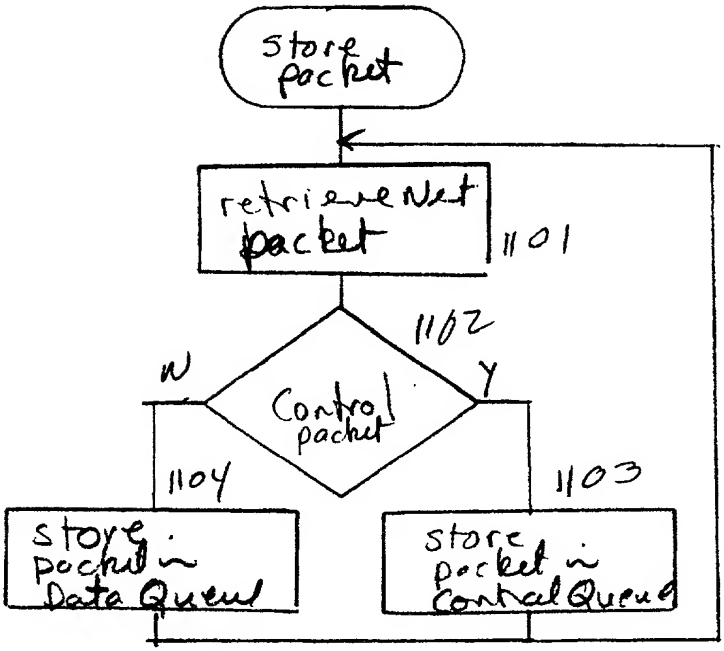

store packet
retrieve Net packet
1101
1102
Control packet
N
Y
1104
1103
store packet in Data Queue
store packet in Control Queue

Fig 11

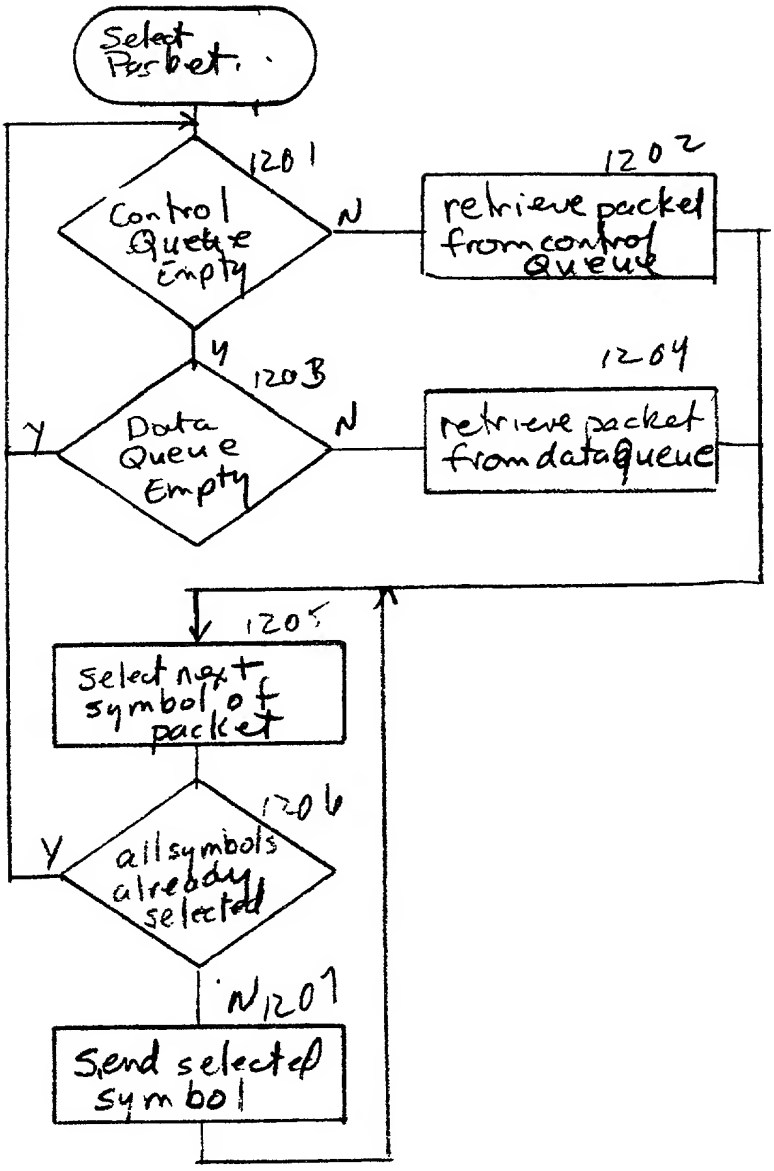

Select Packet
1201
Control Queue Empty
N
1202
retrieve packet from control Queue
Y
1203
Data Queue Empty
Y
N
1204
retrieve packet from data Queue
1205
Select next symbol of packet
1206
all symbols already selected
Y
N 1207
Send selected symbol

Fig 12

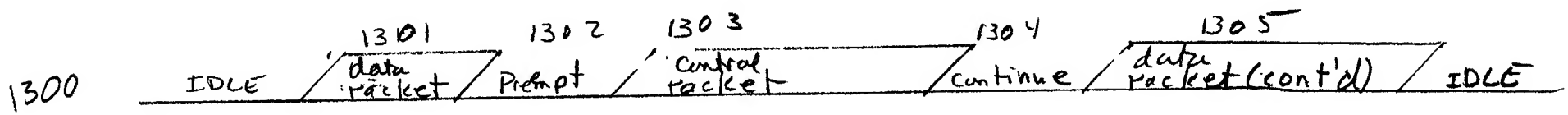

1300
IDLE
1301
data packet
1302
Prompt
1303
control packet
1304
continue
1305
data packet (cont'd)
IDLE

Fig 13

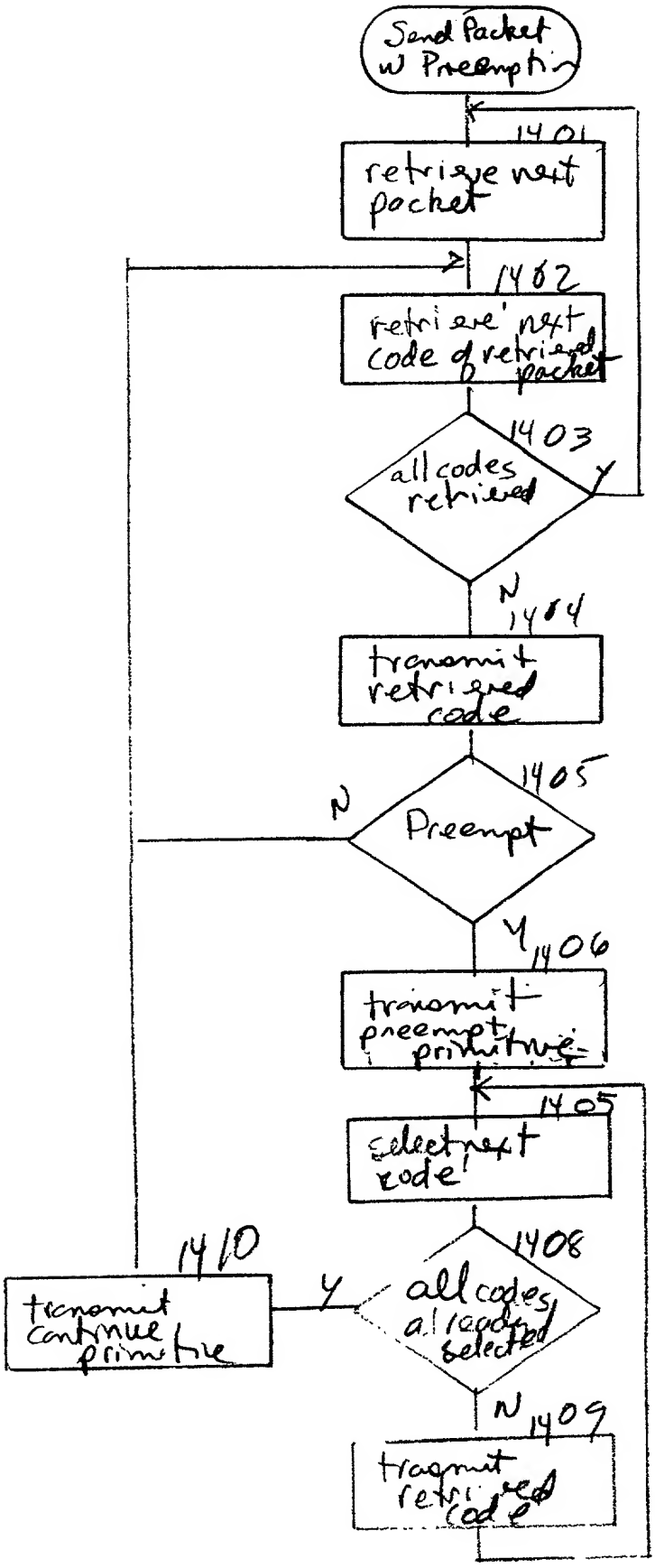

Send Packet w Preempt
1401
retrieve next packet
1402
retrieve next code of retrieved packet
1403
all codes retrieved
Y
N
1404
transmit retrieved code
1405
Preempt
N
Y
1406
transmit preempt primitive
1405
select next code
1408
all codes already selected
Y
1410
transmit continue primitive
N
1409
transmit retrieved code

Fig 14

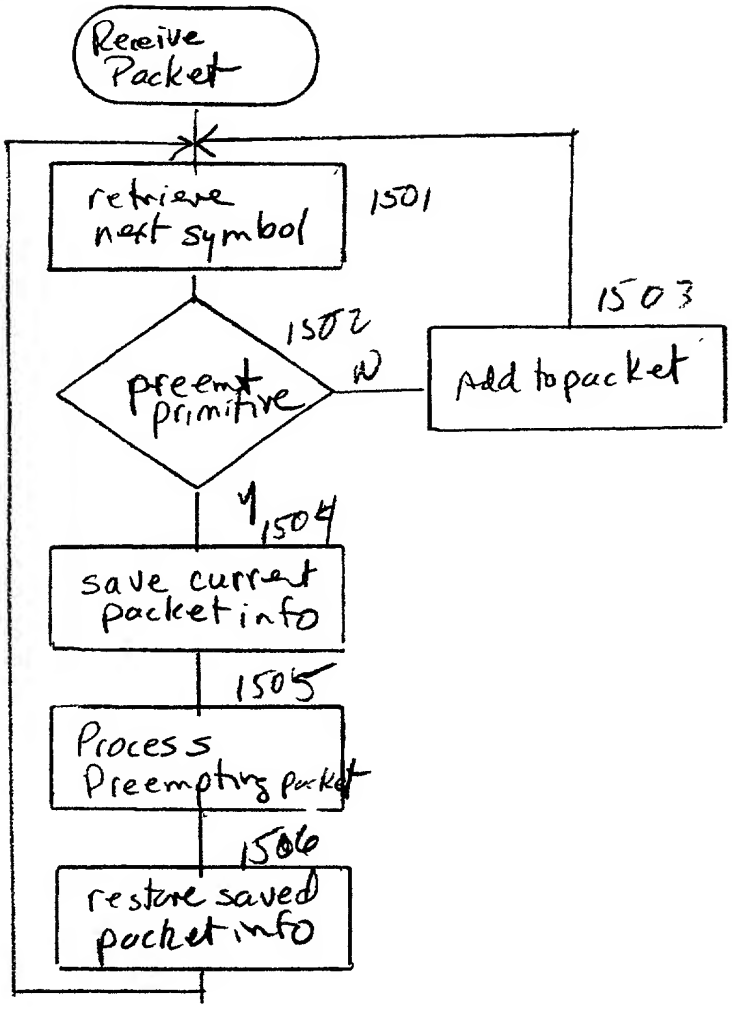

Receive Packet
retrieve next symbol
1501
1502
preempt primitive
N
Add to packet
1503
Y
1504
save current packet info
1505
Process Preempting packet
1506
restore saved packet info

Fig 15

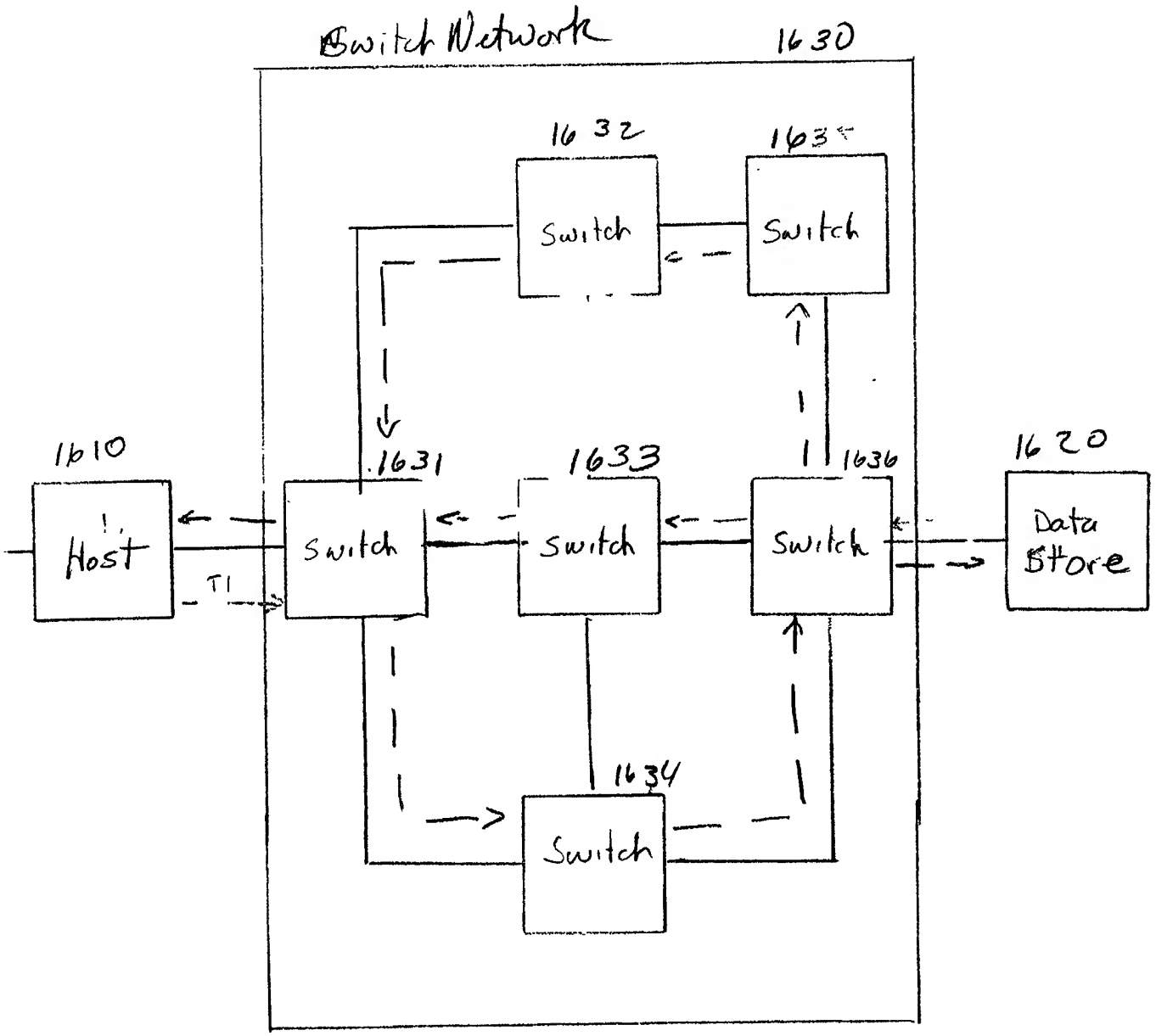

Switch Network
1630
1632
1635
Switch
Switch
1610
1631
1633
1636
1620
Host
Switch
Switch
Switch
Data Store
T1
1634
Switch

Fig 16

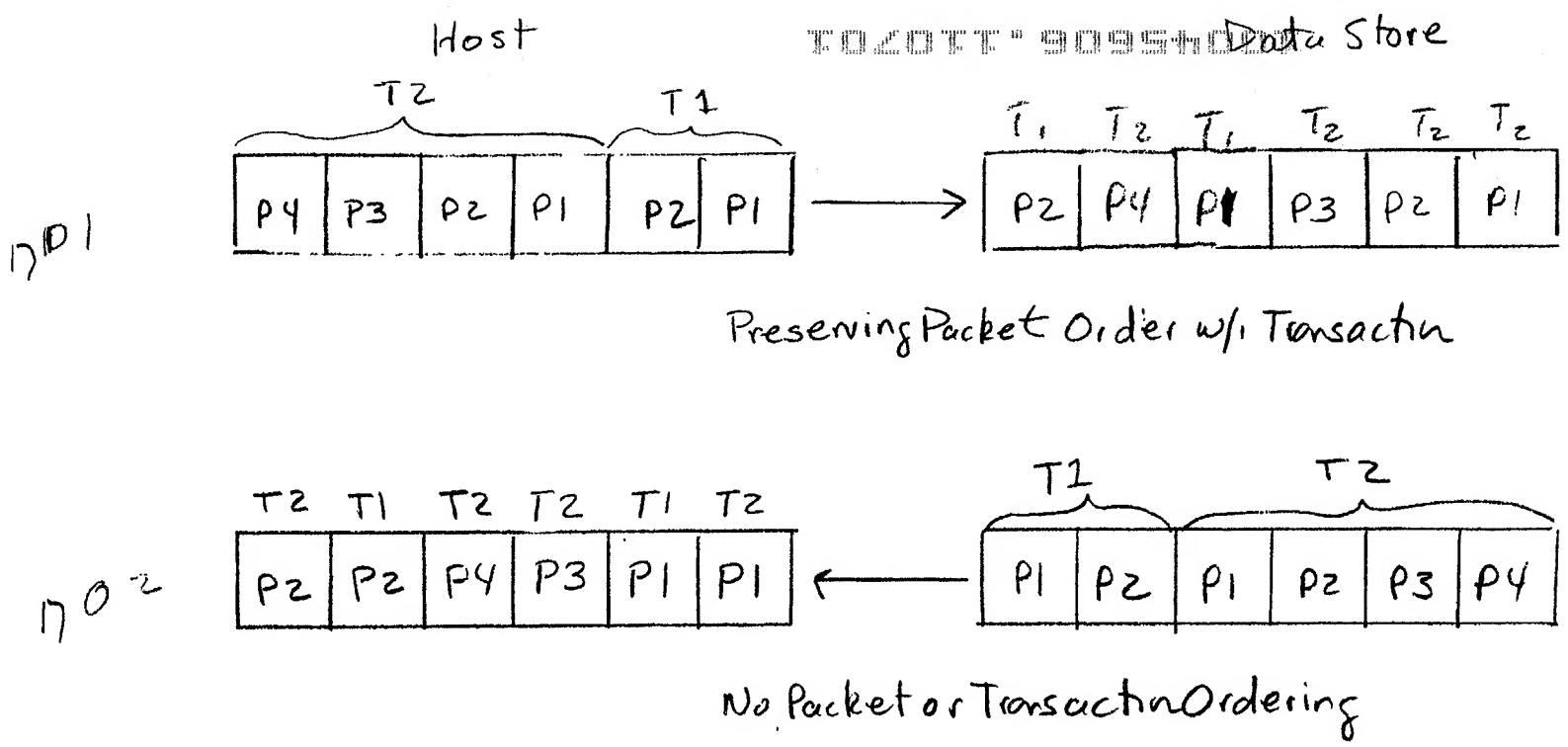

Host
Data Store
T2
T1
P4
P3
P2
P1
P2
P1
T1
T2
T1
T2
T2
T2
P2
P4
P1
P3
P2
P1
Preserving Packet Order w/ Transactn
T2
T1
T2
T2
T1
T2
P2
P2
P4
P3
P1
P1
T1
T2
P1
P2
P1
P2
P3
P4
No Packet or Transactn Ordering

Fig 17

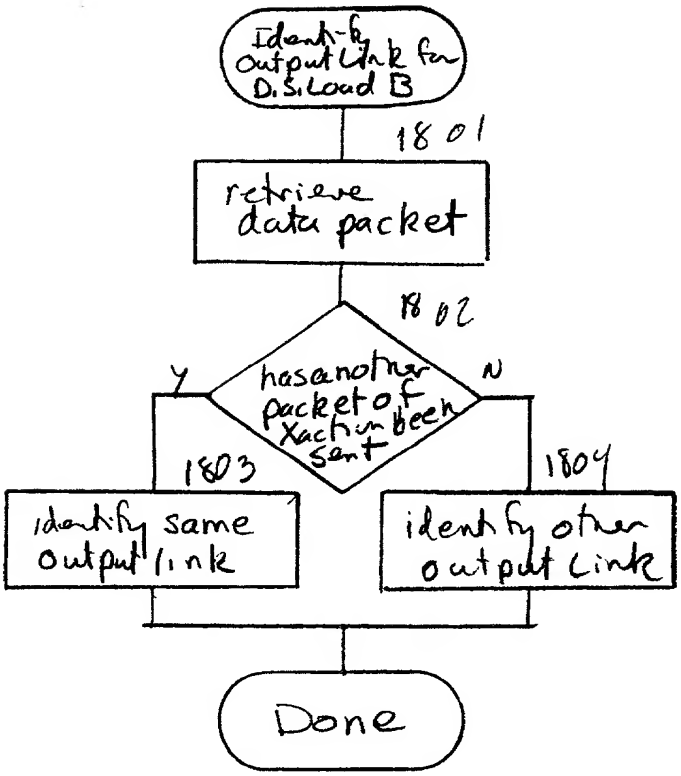

Identify output Link for D.S. Load B
1801
retrieve data packet
1802
has another packet of Xaction been sent
Y
N
1803
1804
identify same output link
identify other output Link
Done

Fig 18

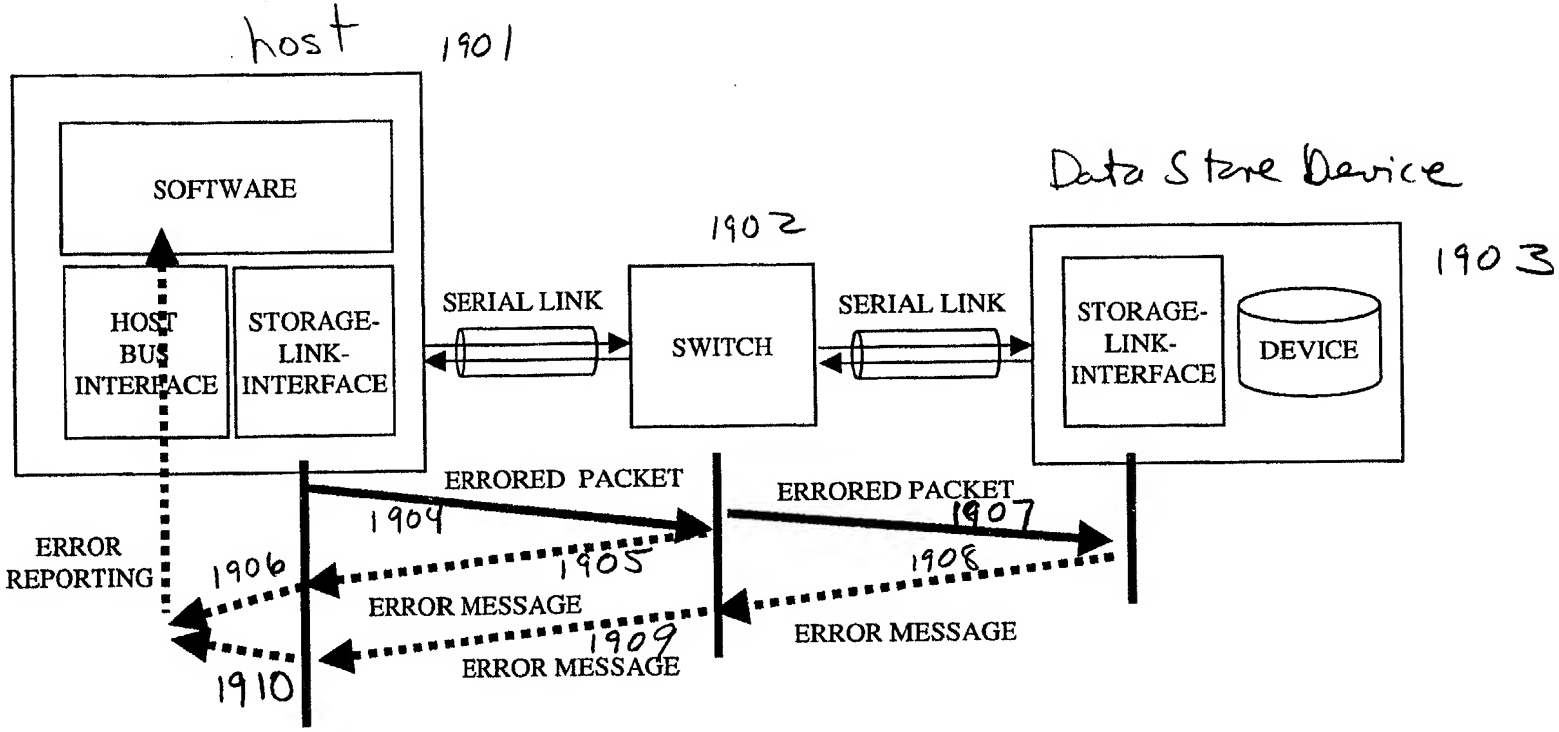

host
1901
SOFTWARE
HOST BUS INTERFACE
STORAGE-LINK-INTERFACE
SERIAL LINK
1902
SWITCH
SERIAL LINK
Data Store Device
1903
STORAGE-LINK-INTERFACE
DEVICE
ERRORED PACKET
1904
ERRORED PACKET
1907
ERROR REPORTING
1906
1905
1908
ERROR MESSAGE
1909
ERROR MESSAGE
ERROR MESSAGE
1910

Fig 19A

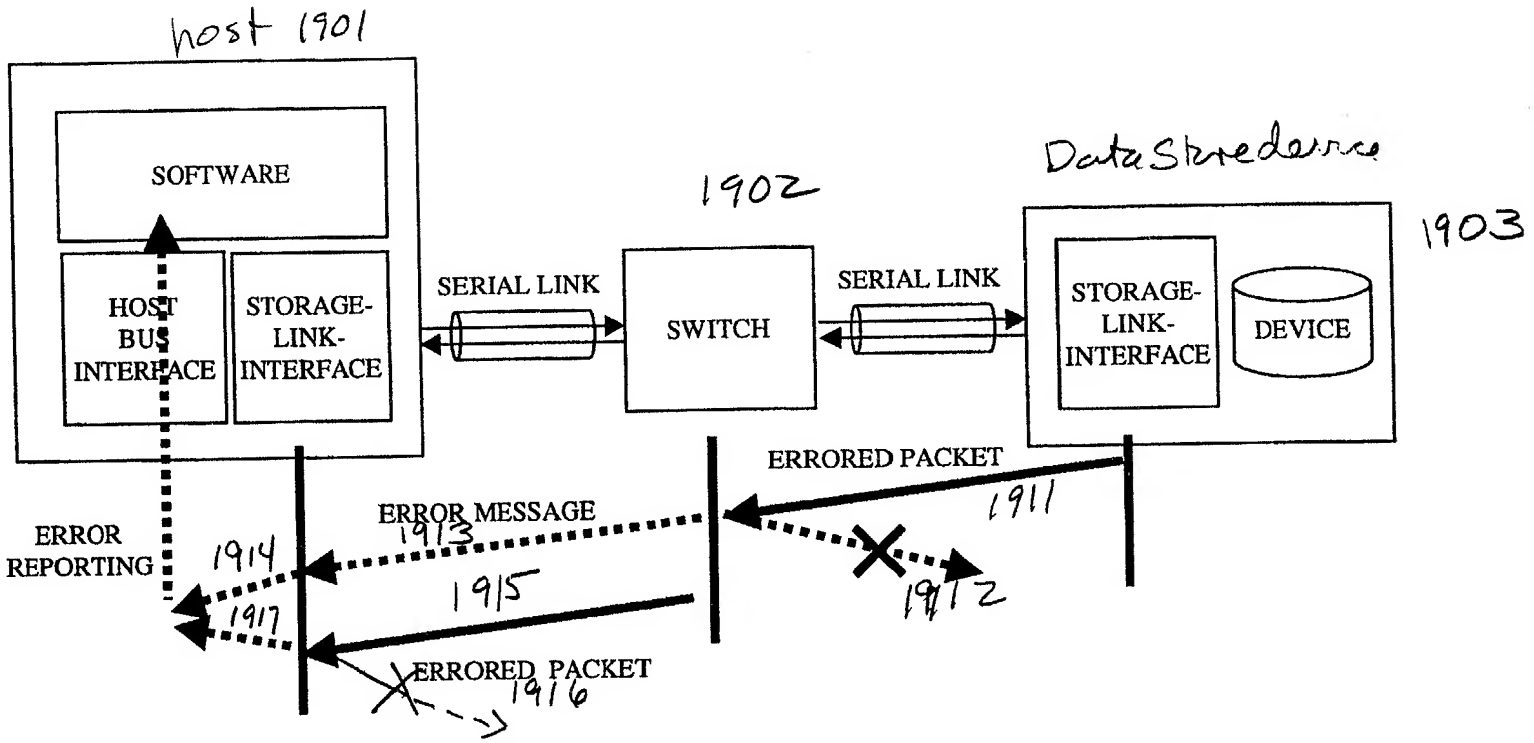
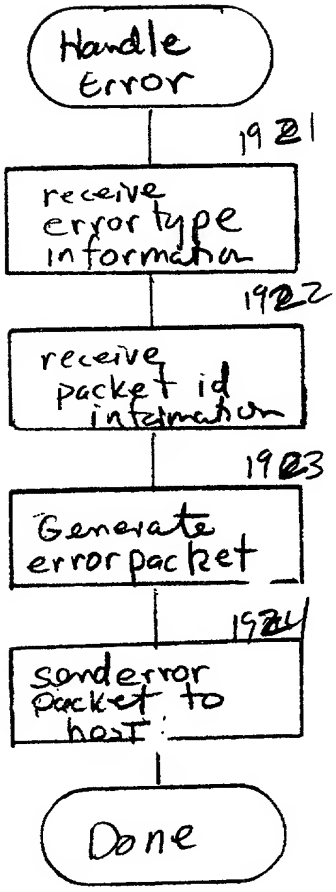


Fig 19B

~~FIG 20~~


Handle Error
1921
receive error type information
1922
receive packet id information
1923
Generate error packet
1924
send error packet to host
Done

19C

8b Code	9 bit symbol
0000 0000	101010101
0000 0001	101010100
0000 0010	101010111
⋮	
0101 0101	001010101
⋮	
0111 0110	001110110
0111 0111	100100010
⋮	
1111 1111	110101010

Fig 20

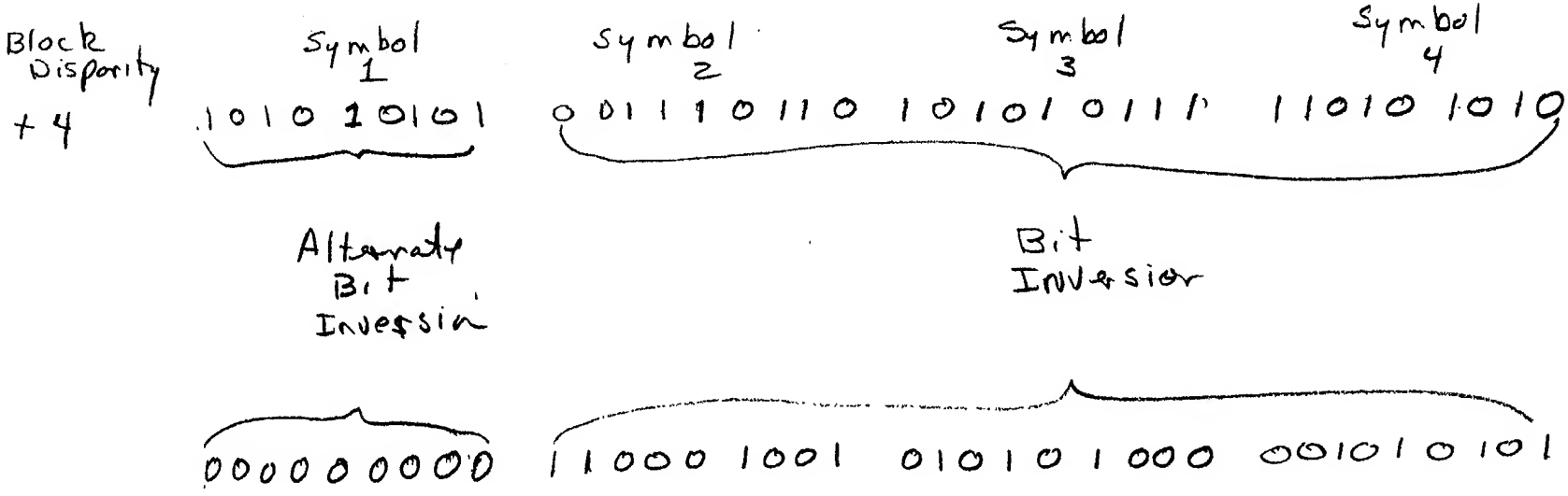

Block Disparity
+4
Symbol 1
Symbol 2
Symbol 3
Symbol 4
1010 10101
0 0111 0110 10101 0111
11010 1010
Alternate Bit Inversion
Bit Inversion
0000000000
11000 1001 01010 1000 00101 0101

Fig 21A

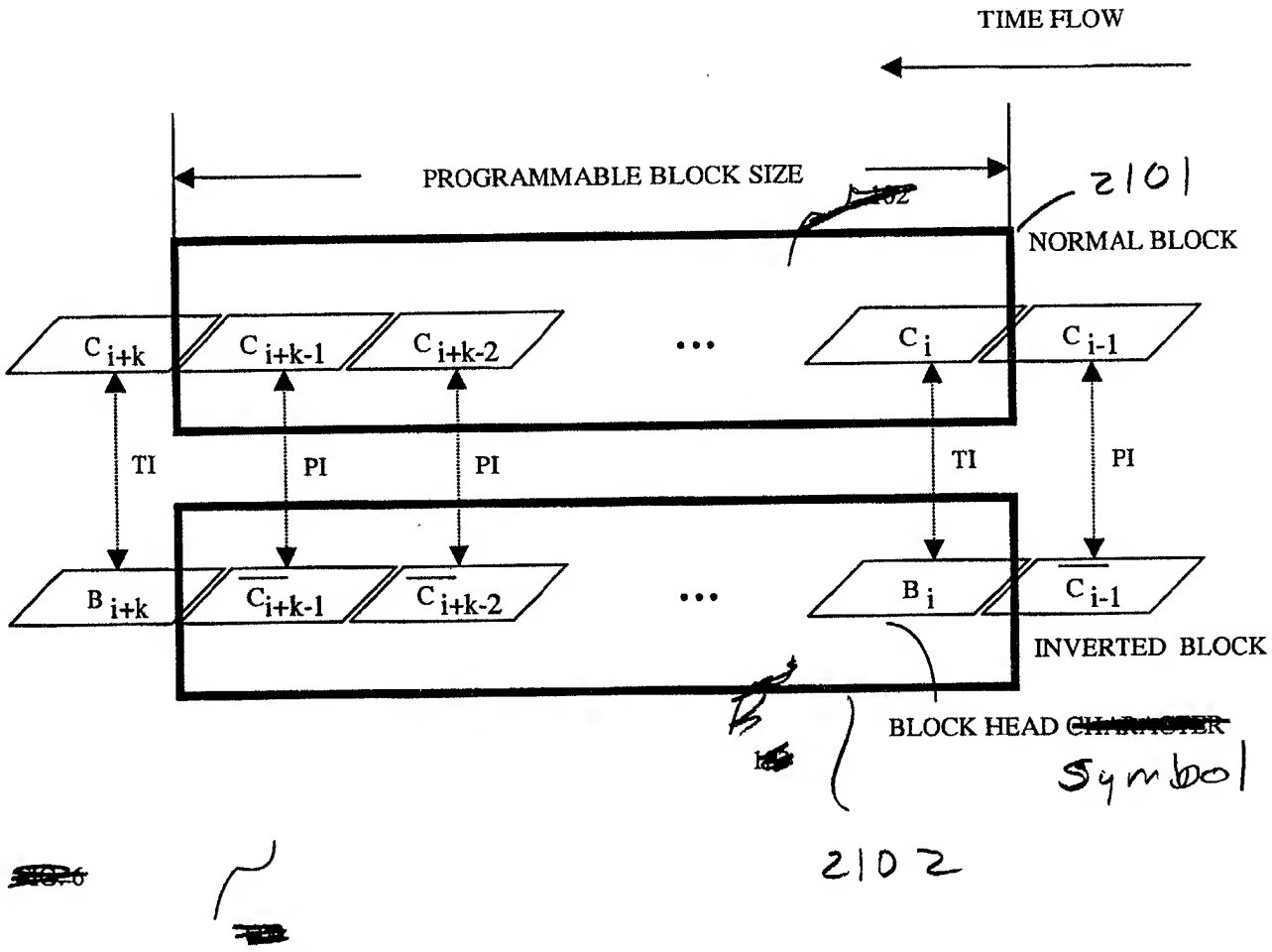

TIME FLOW
PROGRAMMABLE BLOCK SIZE
2101
NORMAL BLOCK
C i+k
C i+k-1
C i+k-2
C i
C i-1
TI
PI
PI
TI
PI
B i+k
C i+k-1
C i+k-2
B i
C i-1
INVERTED BLOCK
BLOCK HEAD Symbol
2102

Fig 21B

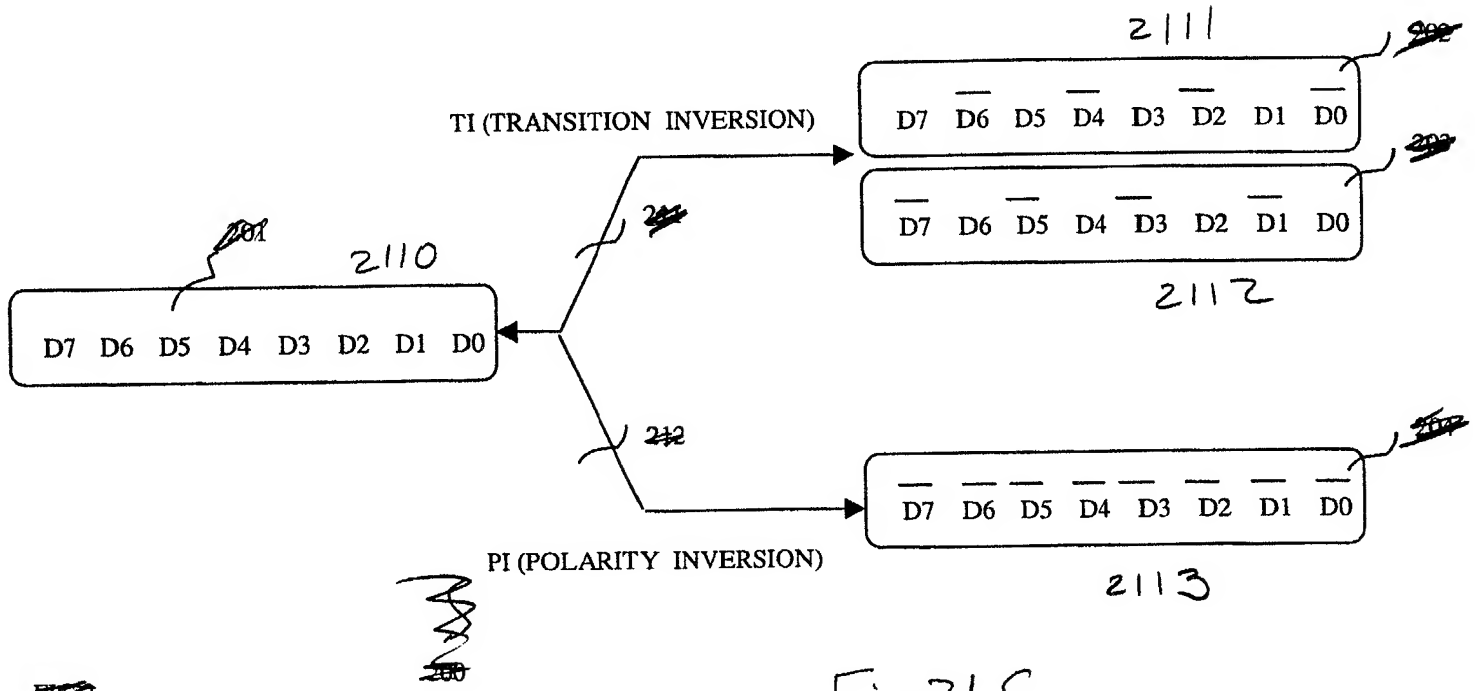

2111
TI (TRANSITION INVERSION)
D7 D6 D5 D4 D3 D2 D1 D0
D7 D6 D5 D4 D3 D2 D1 D0
2112
2110
D7 D6 D5 D4 D3 D2 D1 D0
D7 D6 D5 D4 D3 D2 D1 D0
PI (POLARITY INVERSION)
2113

FIG.2

Fig 21C

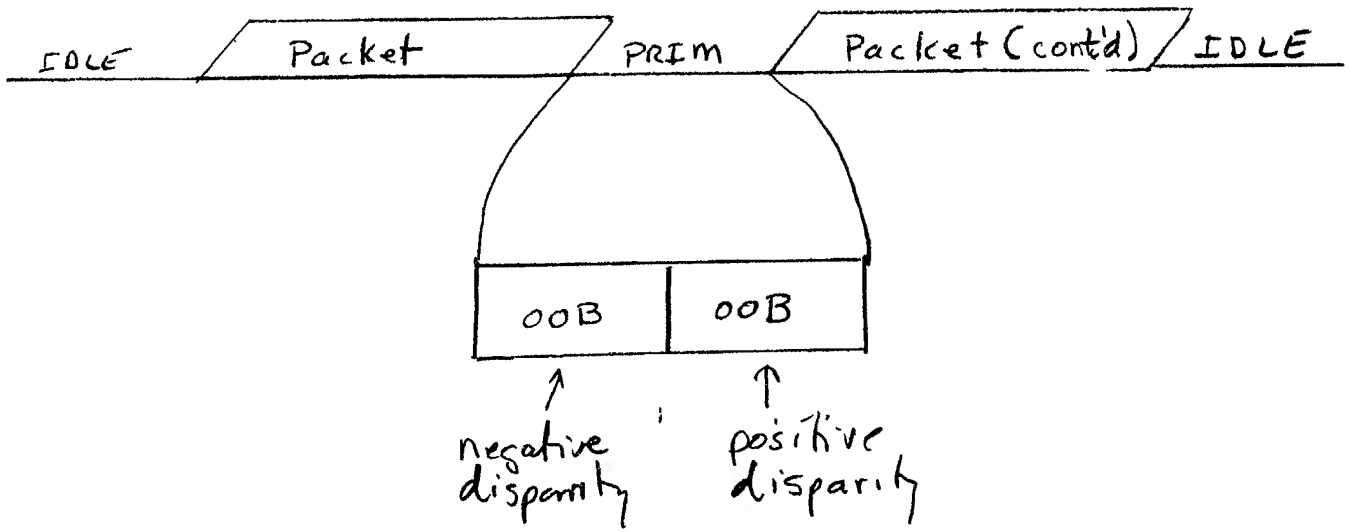

IDLE
Packet
PRIM
Packet (cont'd)
IDLE
OOB
OOB
negative disparity
positive disparity

Fig 22

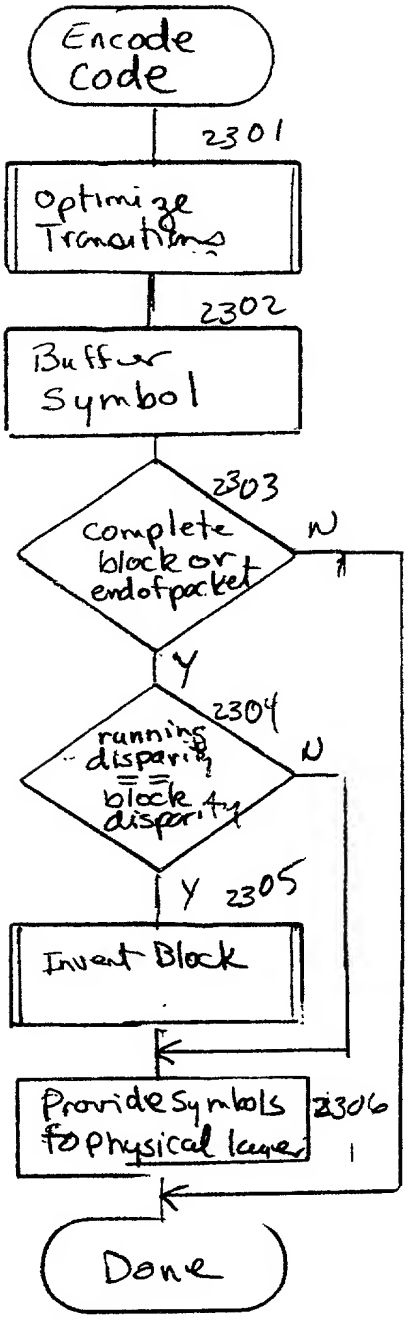

Encode Code
2301
Optimize Transitions
2302
Buffer Symbol
2303
complete block or endofpacket
N
Y
2304
running disparity == block disparity
N
Y
2305
Invert Block
Provide Symbols to physical layer
2306
Done

Fig 23

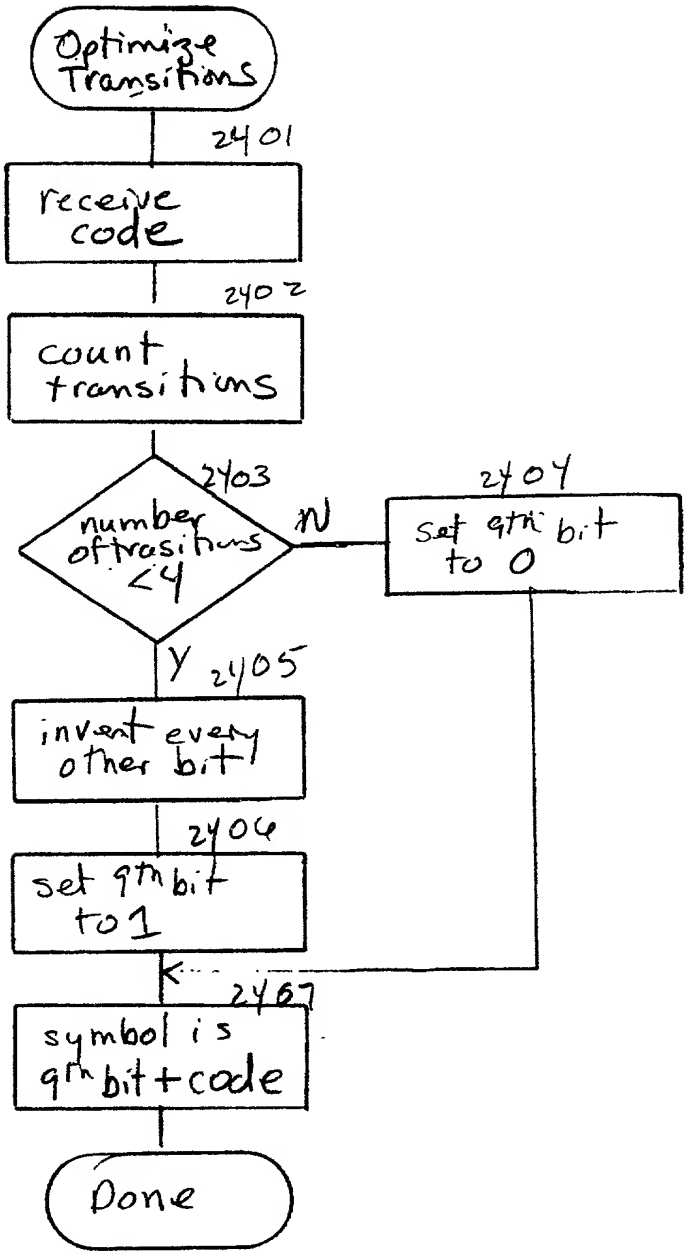

Optimize Transitions
2401
receive code
2402
count transitions
2403
number of trasitions <4
N
2404
set qth bit to 0
Y
2405
invert every other bit
2406
set qth bit to 1
2407
symbol is qth bit + code
Done

Fig 24

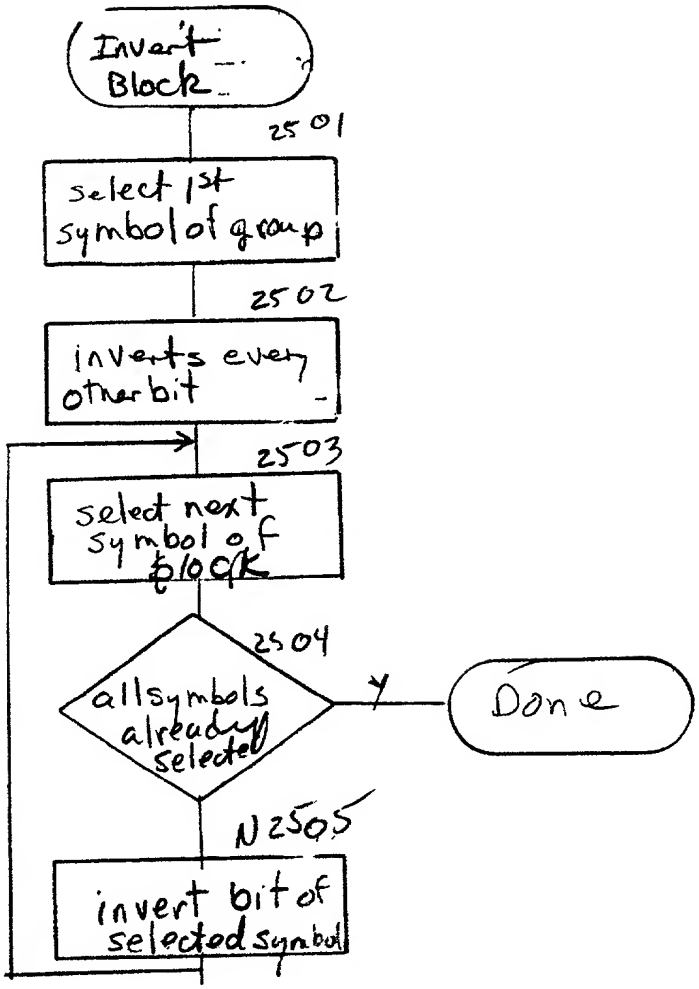

Invert Block
2501
select 1st symbol of group
2502
inverts every other bit
2503
select next symbol of block
2504
all symbols already selected
Y
Done
N
2505
invert bit of selected symbol

Fig 25

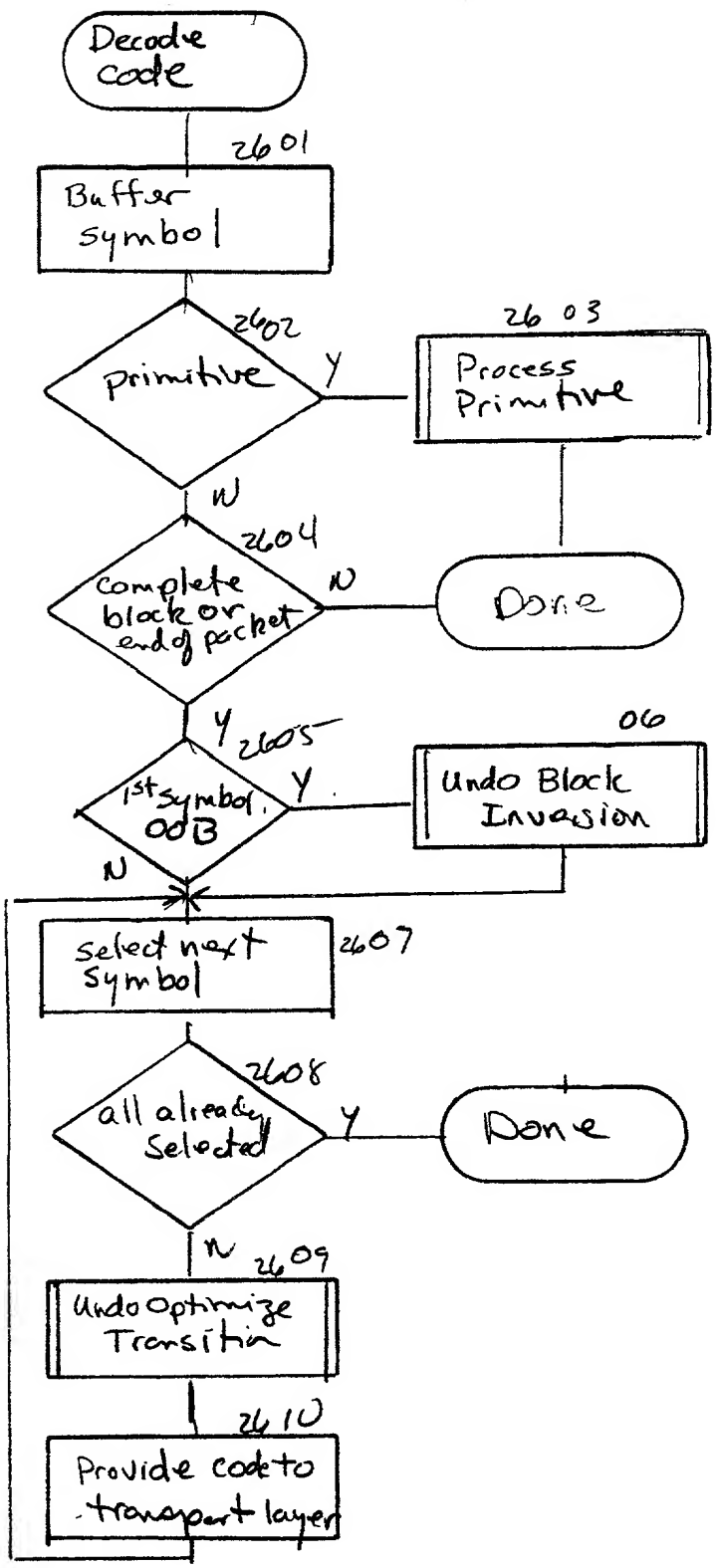

Decode code
2601
Buffer symbol
2602
Primitive
Y
2603
Process Primitive
N
2604
Complete block or end of packet
N
Done
Y
2605
1st symbol ODB
Y
06
Undo Block Invasion
N
Select next symbol
2607
2608
all already Selected
Y
Done
N
2609
Undo Optimize Transition
2610
Provide code to transport layer

Fig 26

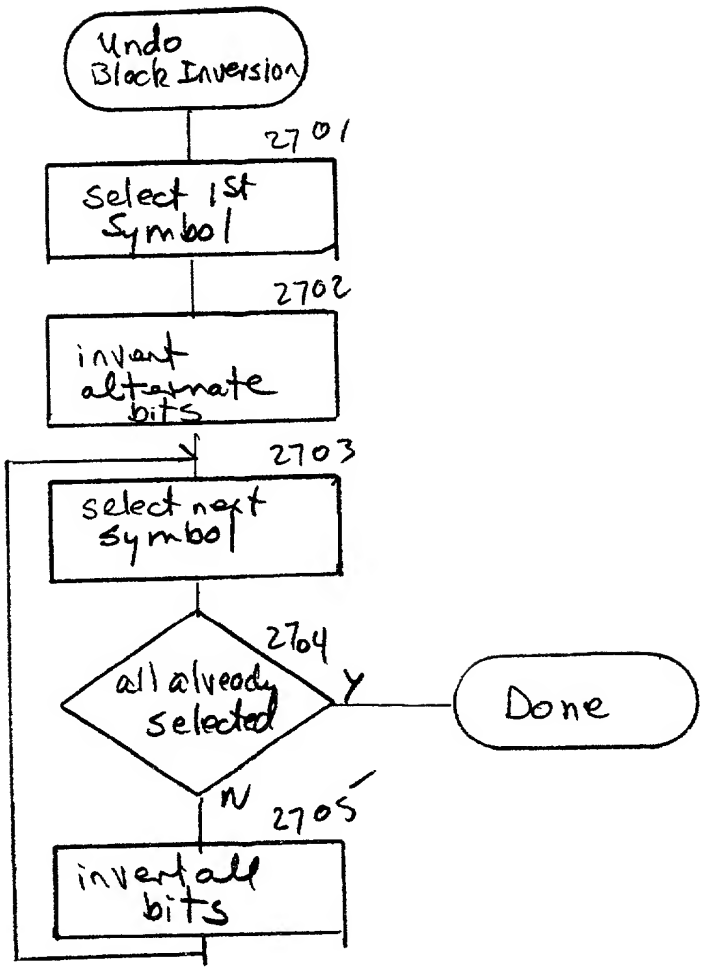

Undo Block Inversion
2701
Select 1st Symbol
2702
invert alternate bits
2703
select next symbol
2704
all already selected
Y
Done
N
2705
invert all bits

Fig 27

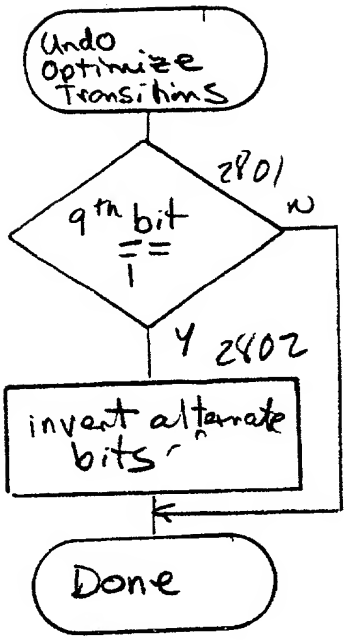

Undo Optimize Transitions
9th bit == 1
2801
N
Y
2802
invert alternate bits
Done

Fig 28

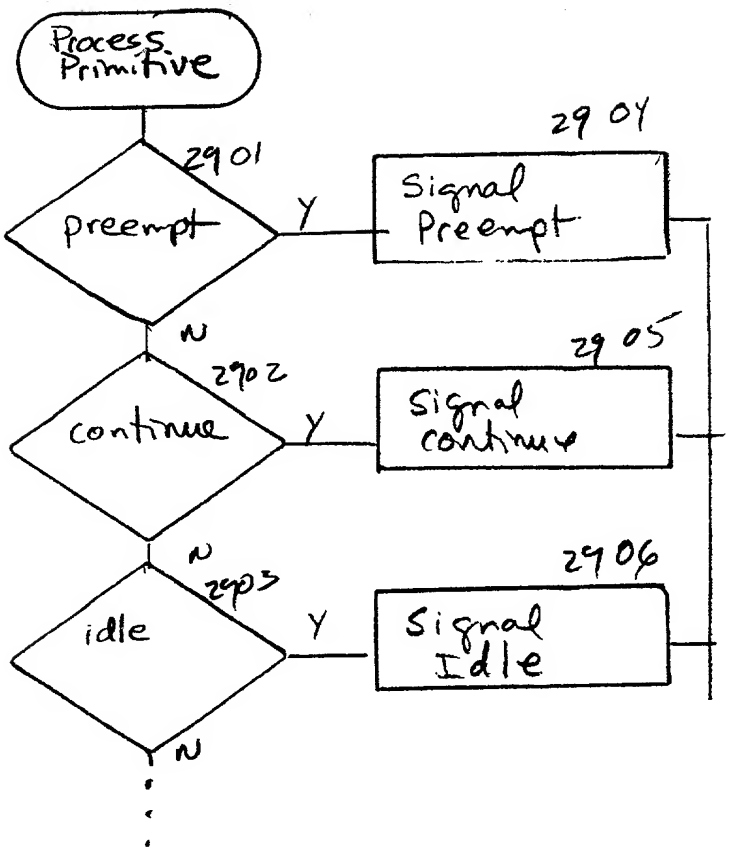

Process Primitive
2901
preempt
Y
2904
Signal Preempt
N
2902
continue
Y
2905
Signal continue
N
2903
idle
Y
2906
Signal Idle
N

Fig 29

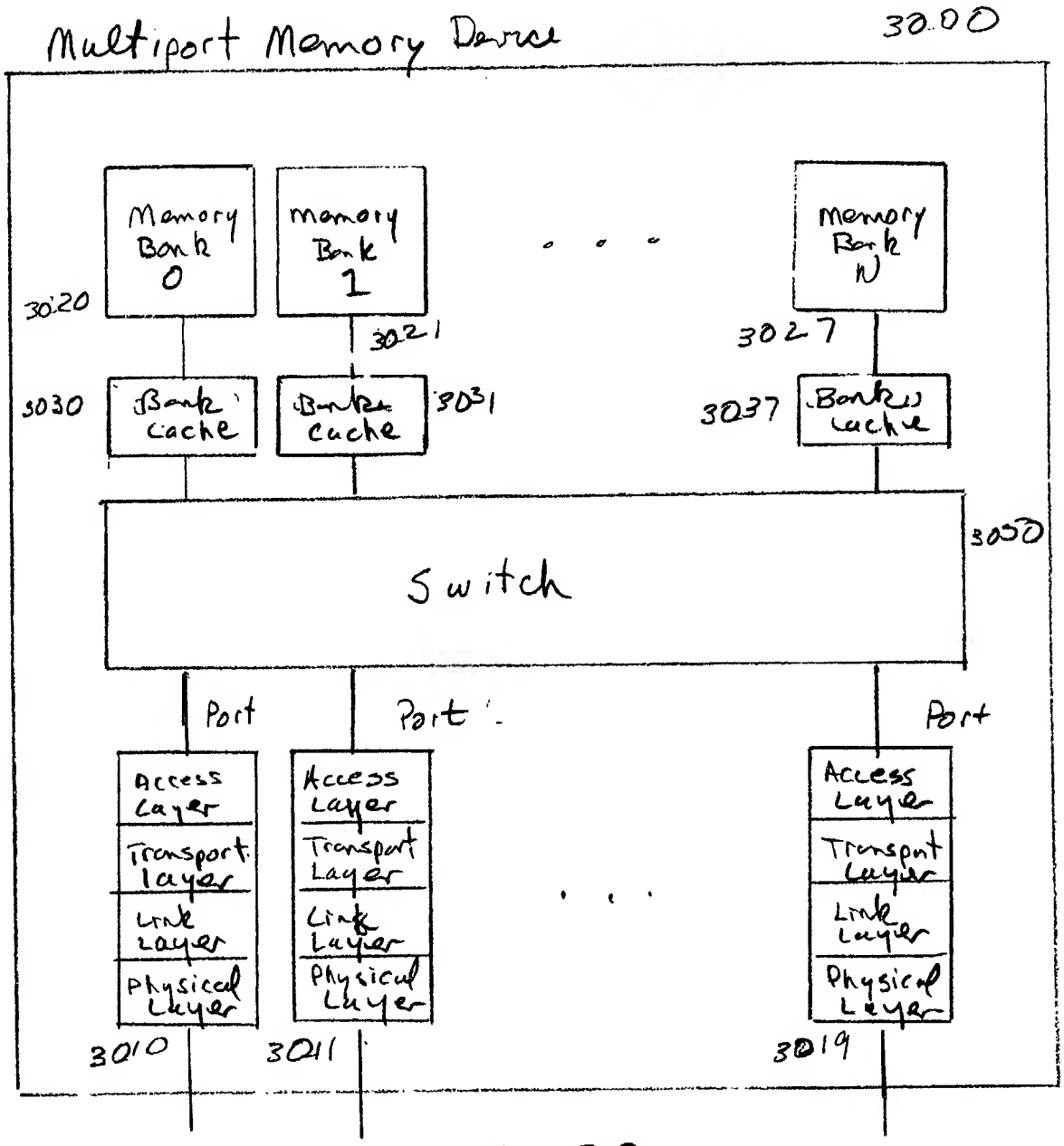


Fig 30

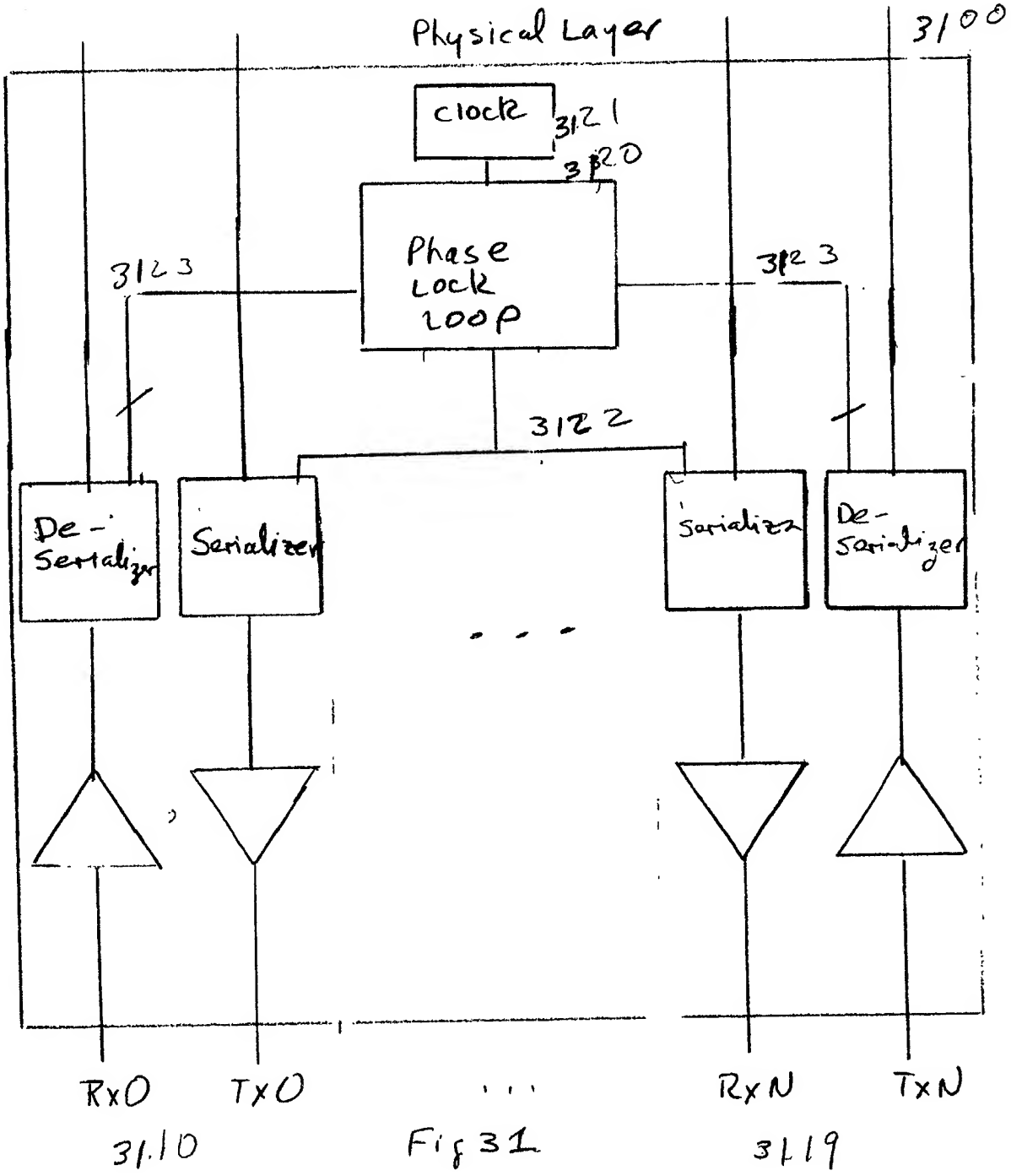


Fig 31

Input Queue 3201

Port	R/W	Address	Data
3	R	1000	
4	W	4000	10....1
3	W	1000	111...0
3	R	2000	
	⋮		

Output Queue 3202

Valid	Port	Data
1	3	11...0
0		
0		
1	3	101...1
	⋮	

Fig 32

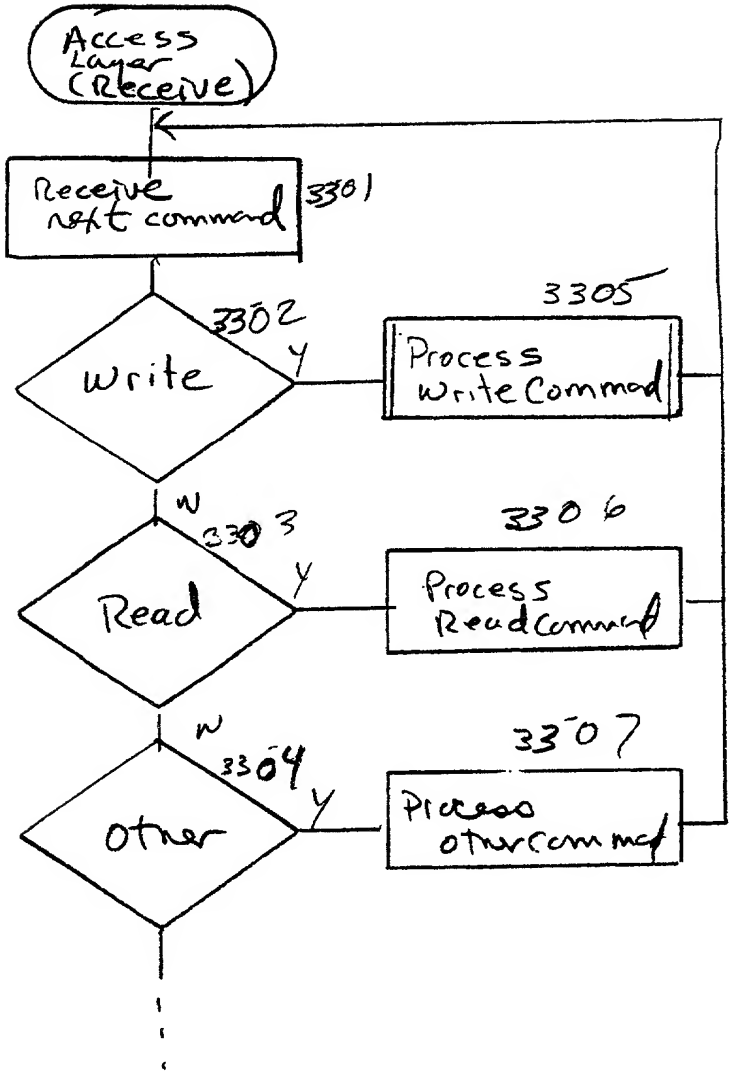

Access Layer (Receive)
Receive next command
3301
Write
3302
Y
Process Write Command
3305
N
Read
3303
Y
Process Read Command
3306
N
Other
3304
Y
Process other command
3307

Fig 33

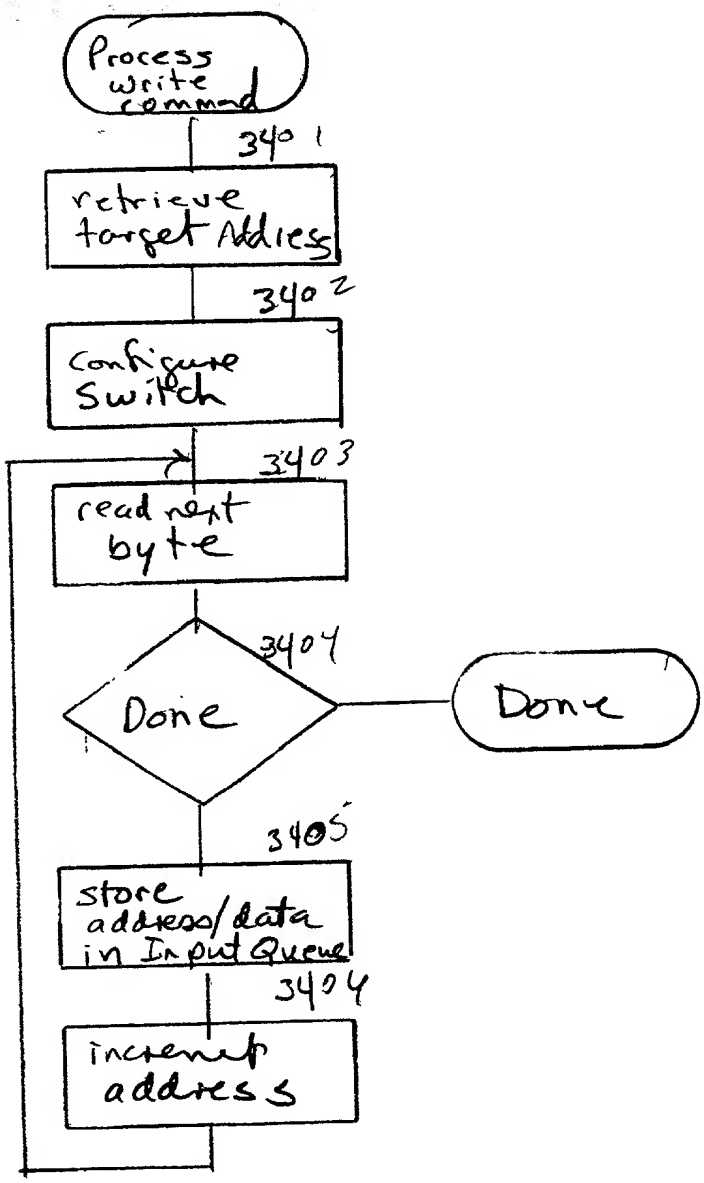

Process write command
3401
retrieve target Address
3402
configure switch
3403
read next byte
3404
Done
Done
3405
store address/data in Input Queue
3406
increment address

Fig 34

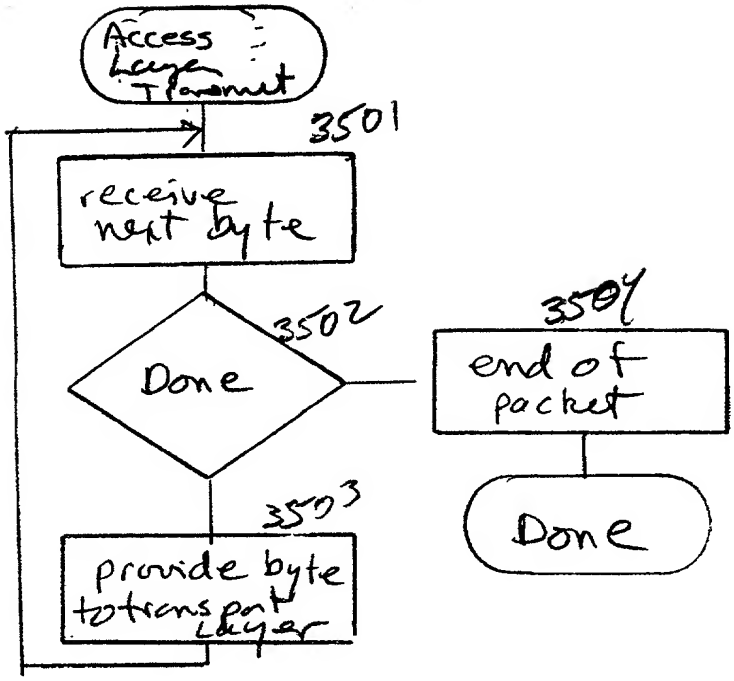

Access Layer Transmit
3501
receive next byte
3502
Done
3504
end of packet
Done
3503
provide byte to transport Layer

Fig 35

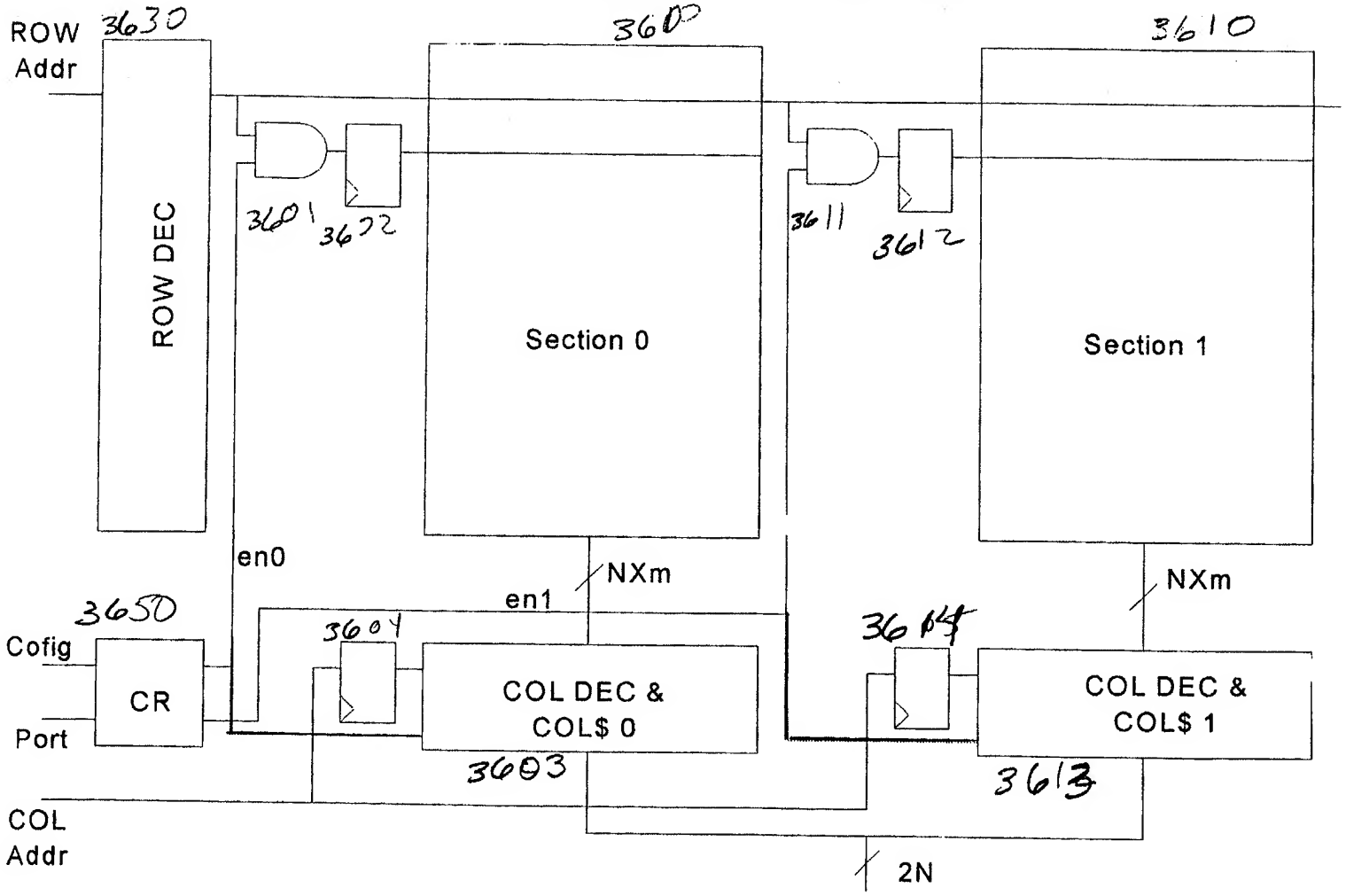
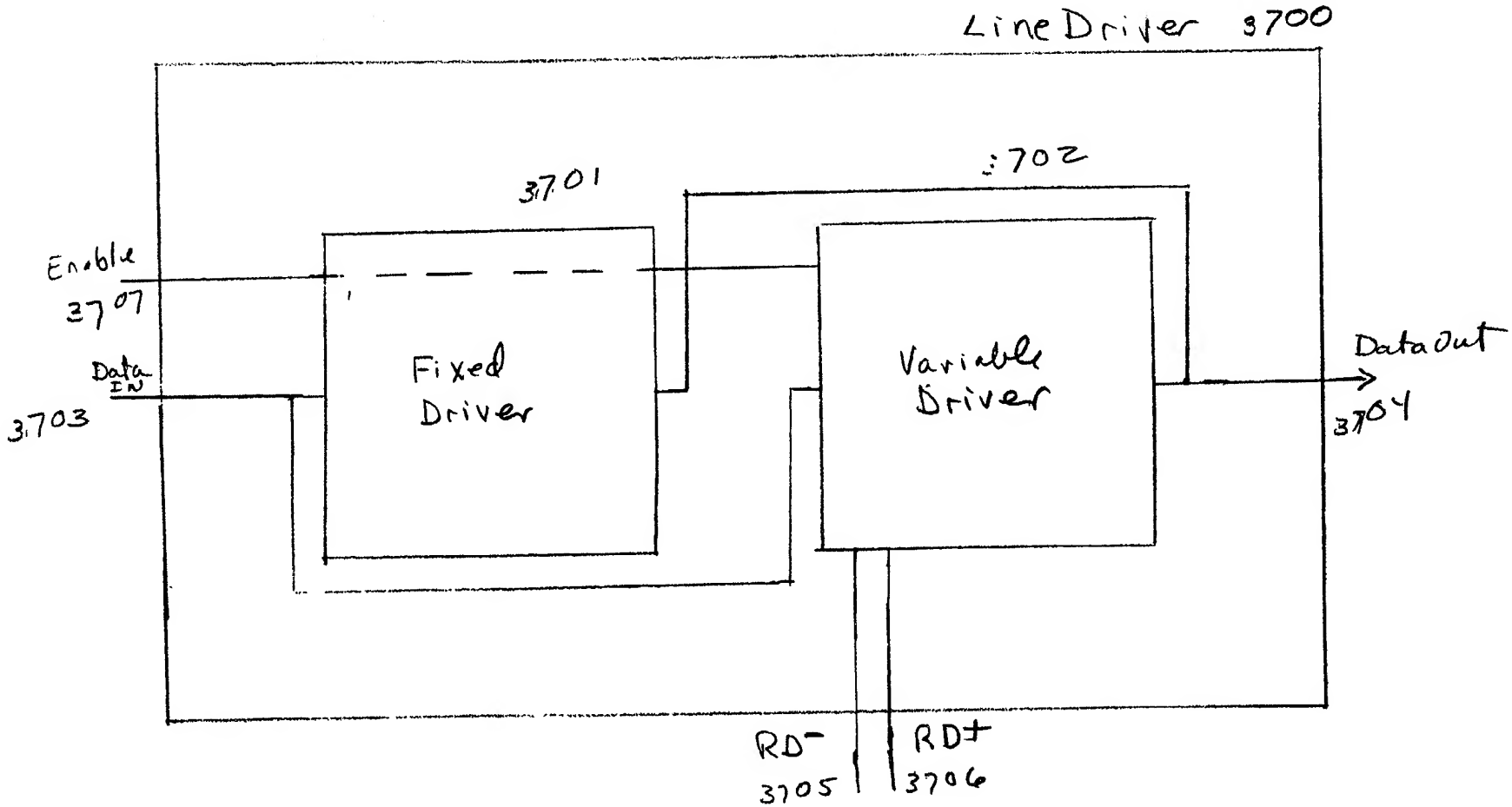


Fig 36



Variable Driver

$\begin{cases} RD^+ \wedge \overline{DataIN} = \text{pull down} \\ RD^- \wedge DataIN = \text{pull up} \end{cases}$

Fig 37A

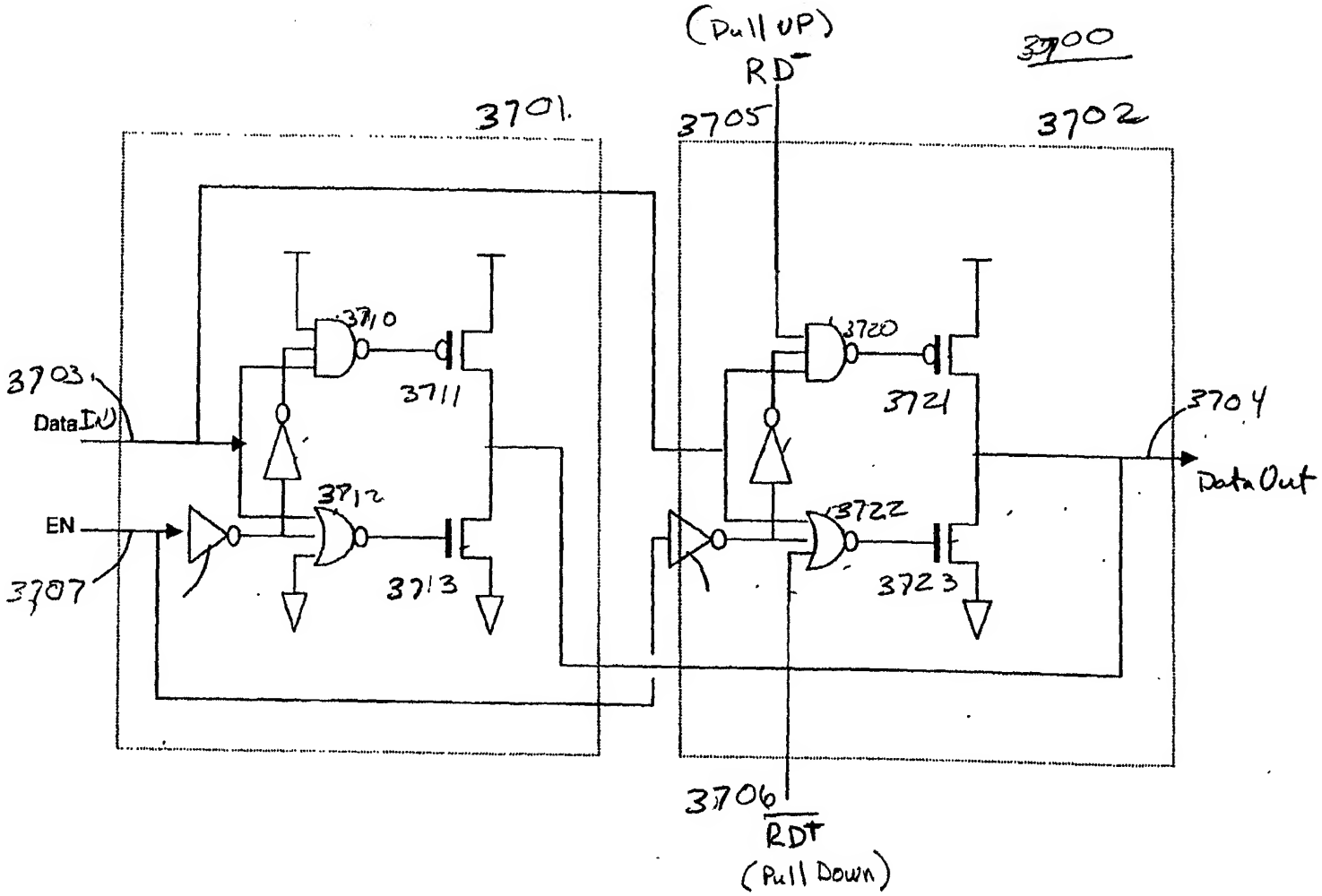


Fig 37B

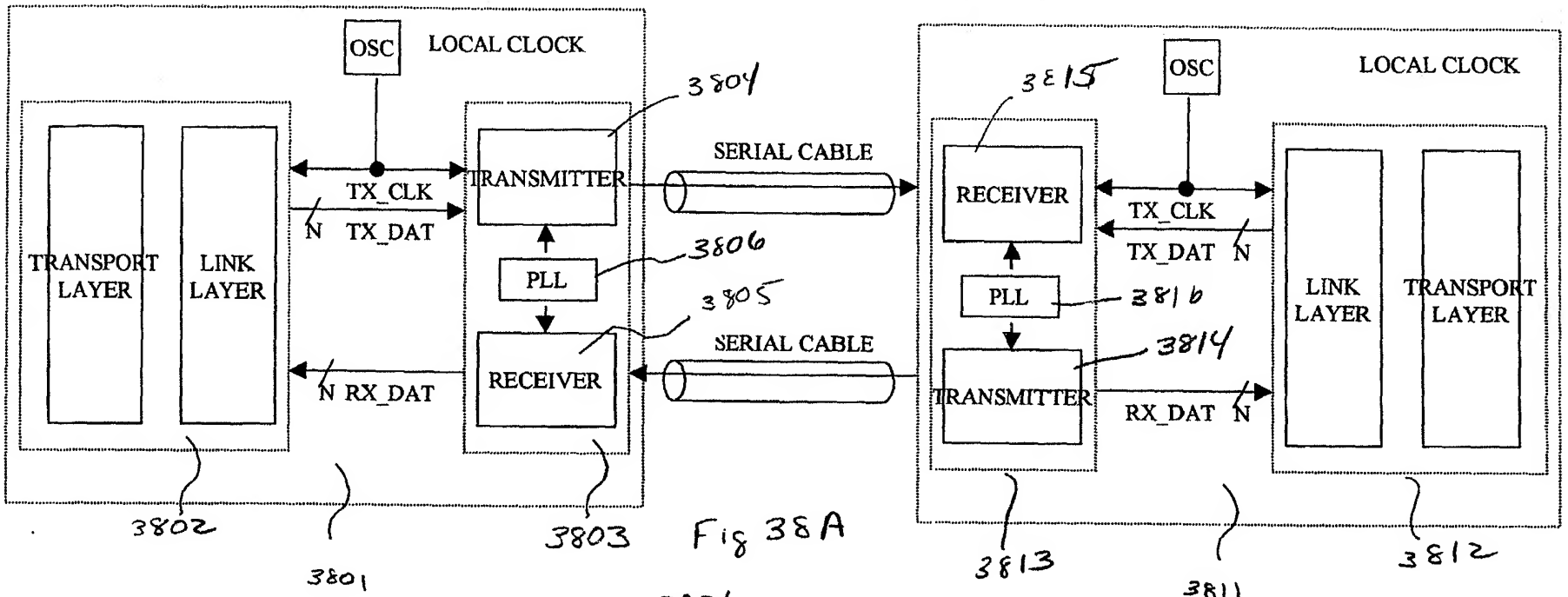


Fig 38A

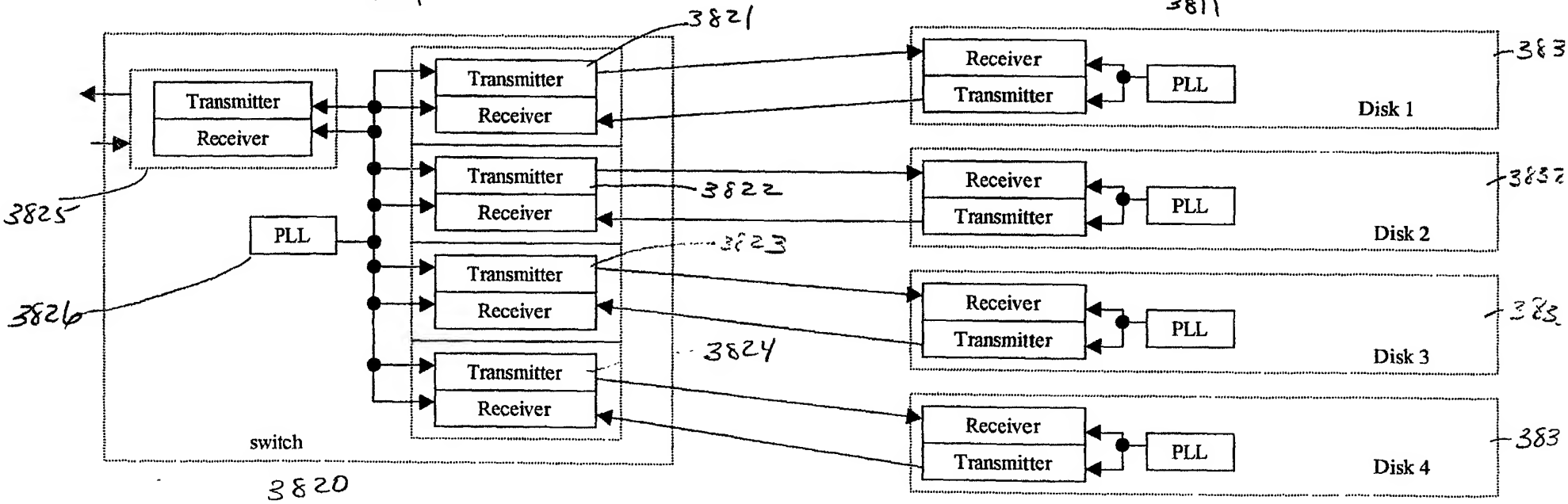


Fig 38B

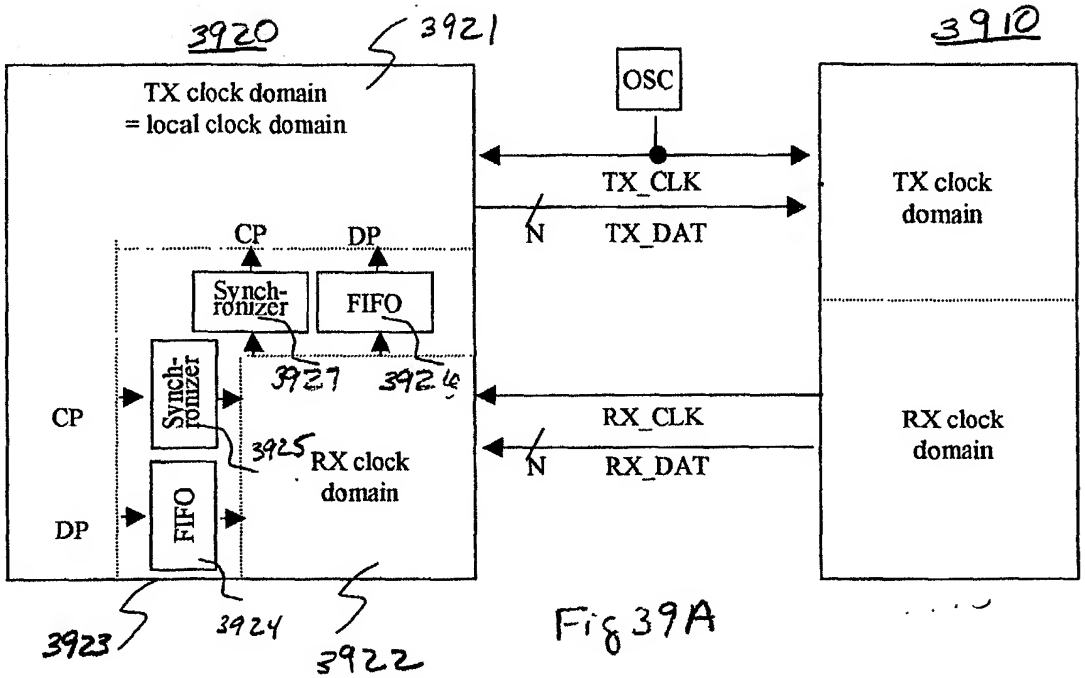


Fig 39A

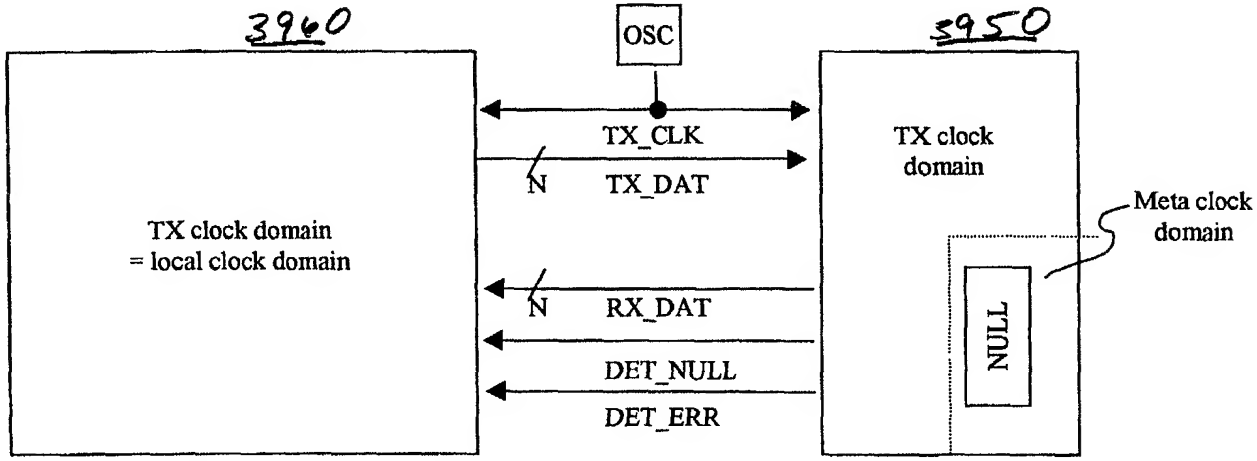


Fig 39B

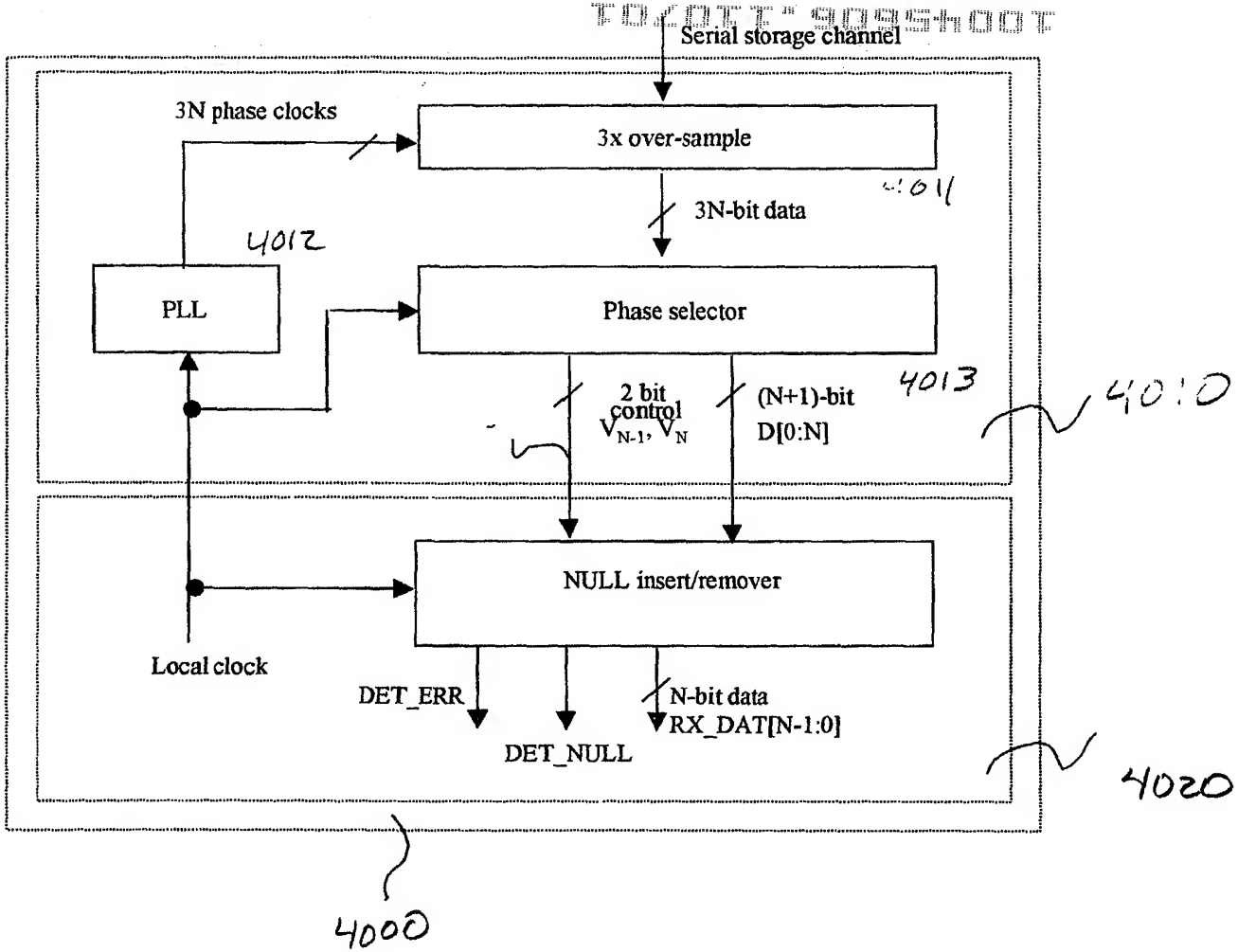

Serial storage channel
3N phase clocks
3x over-sample
4011
3N-bit data
4012
PLL
Phase selector
4013
2 bit control
V_{N-1}, V_N
(N+1)-bit
D[0:N]
4010
NULL insert/remover
Local clock
DET_ERR
DET_NULL
N-bit data
RX_DAT[N-1:0]
4020
4000

Fig 4D

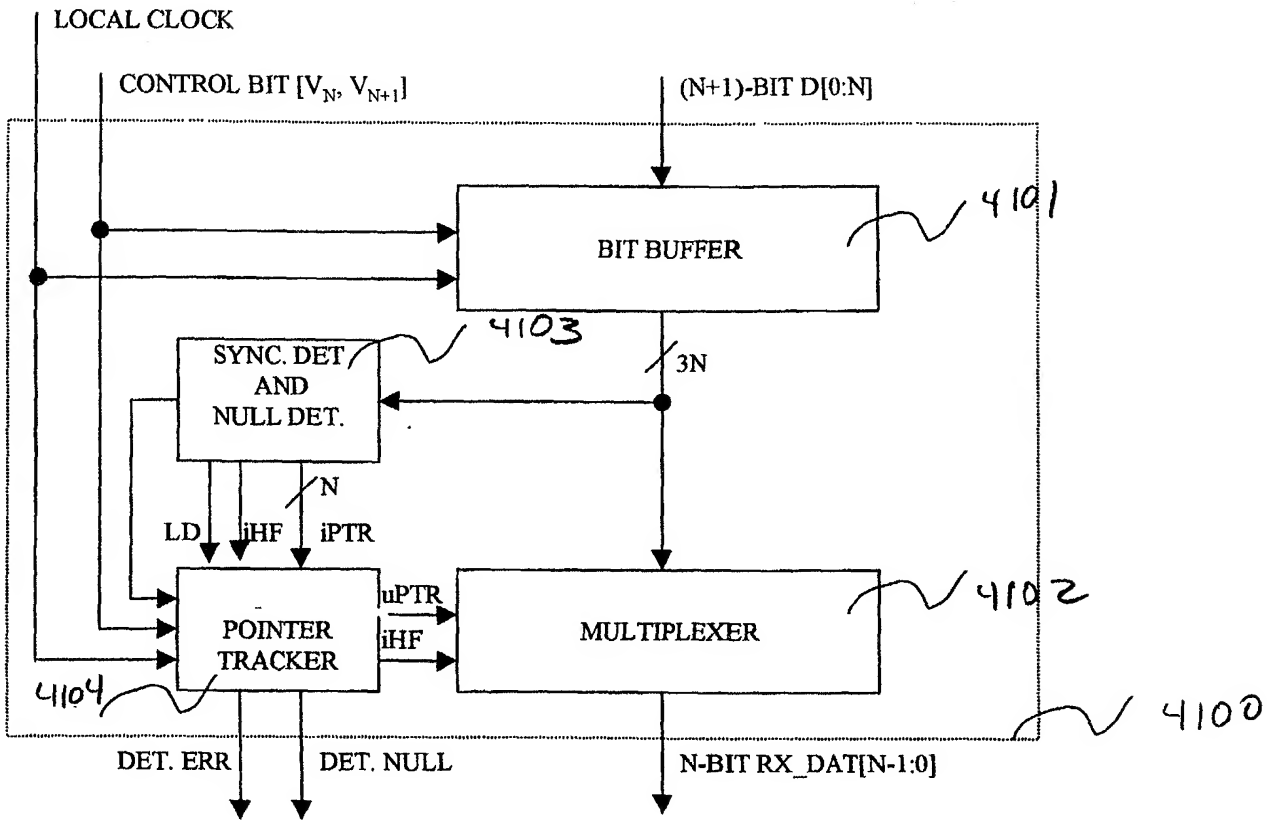

LOCAL CLOCK
CONTROL BIT [V_N, V_{N+1}]
(N+1)-BIT D[0:N]
BIT BUFFER
4101
4103
SYNC. DET AND NULL DET.
3N
N
LD
iHF
iPTR
POINTER TRACKER
uPTR
iHF
MULTIPLEXER
4102
4104
4100
DET. ERR
DET. NULL
N-BIT RX_DAT[N-1:0]

Fig 41

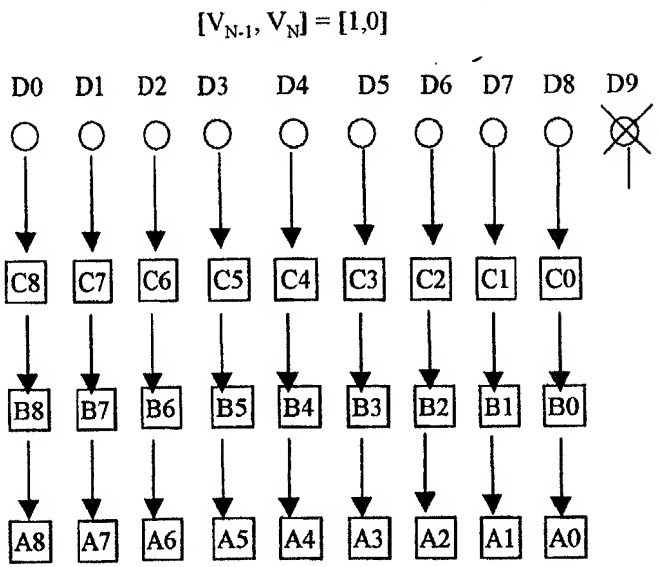

$[V_{N-1}, V_N] = [1,0]$
D0
D1
D2
D3
D4
D5
D6
D7
D8
D9
C8
C7
C6
C5
C4
C3
C2
C1
C0
B8
B7
B6
B5
B4
B3
B2
B1
B0
A8
A7
A6
A5
A4
A3
A2
A1
A0

Fig 42A

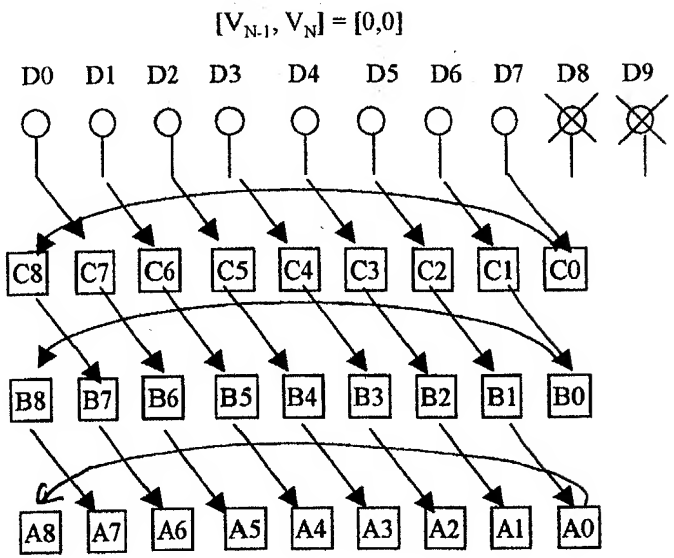


Fig 42B

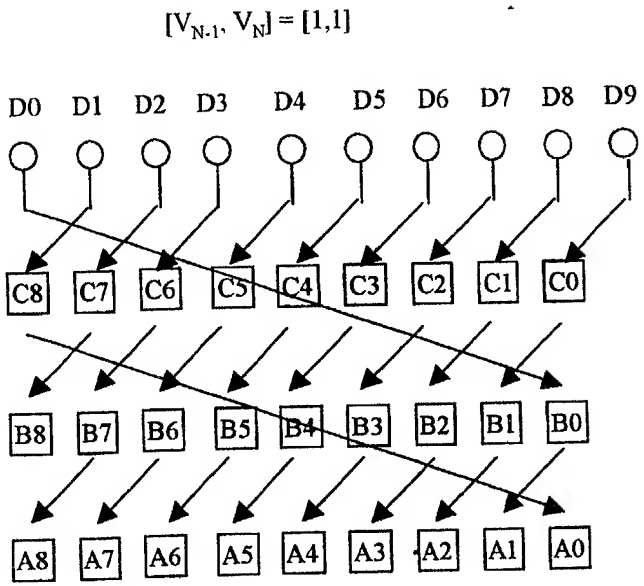

$[V_{N-1}, V_N] = [1,1]$
D0 D1 D2 D3 D4 D5 D6 D7 D8 D9
C8 C7 C6 C5 C4 C3 C2 C1 C0
B8 B7 B6 B5 B4 B3 B2 B1 B0
A8 A7 A6 A5 A4 A3 A2 A1 A0

Fig 42C

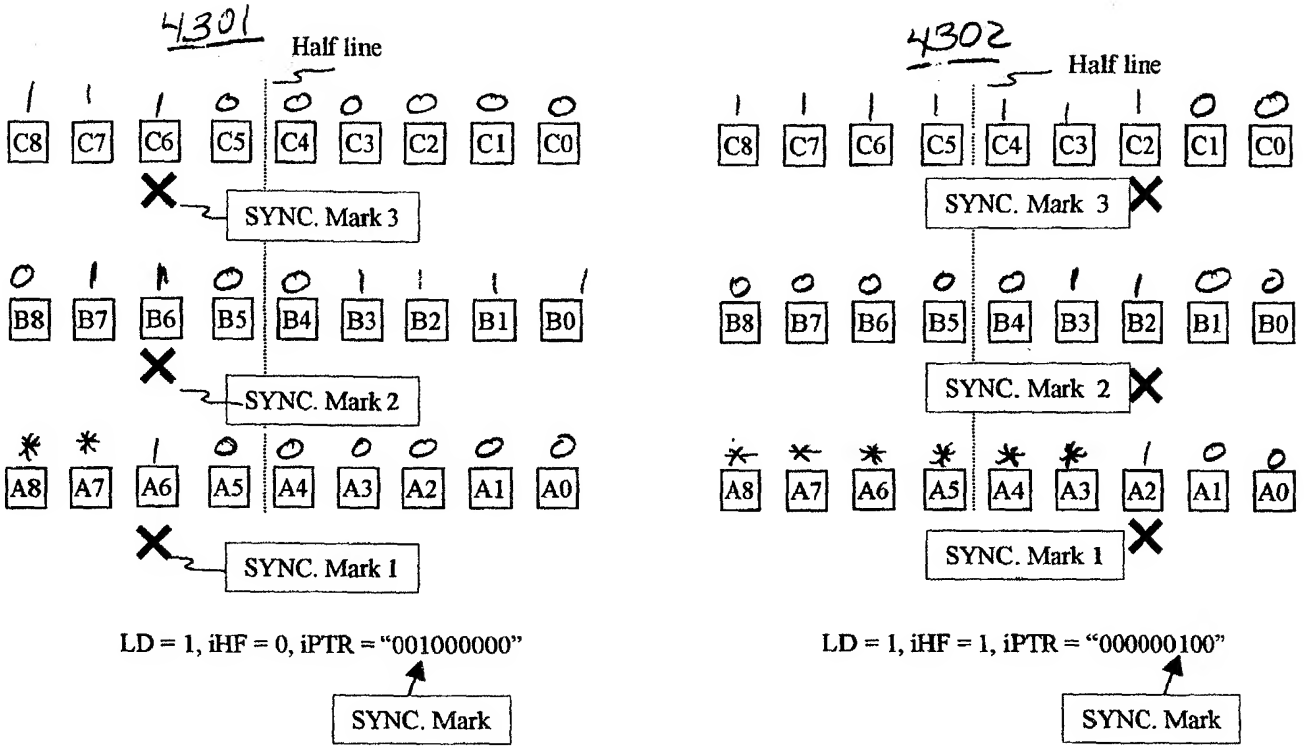
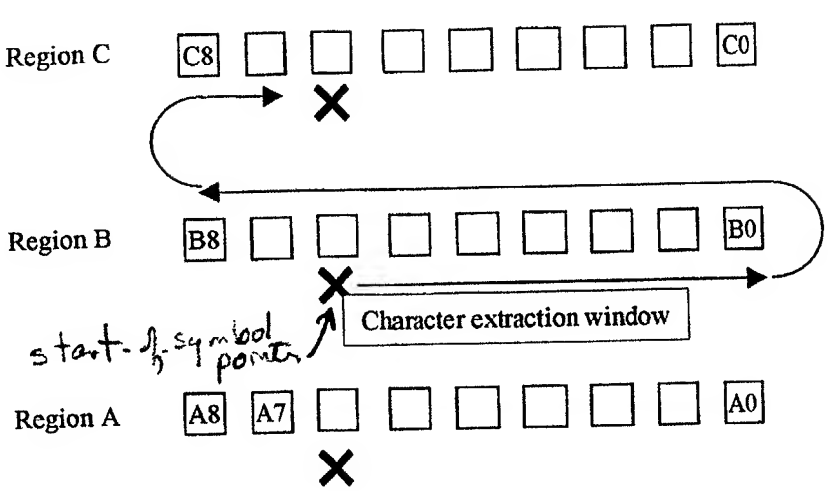
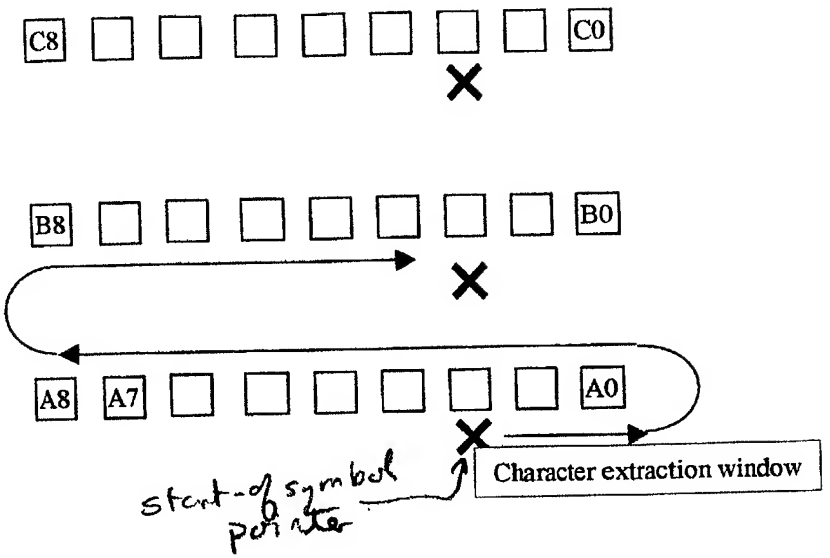

4301
Half line
C8 C7 C6 C5 C4 C3 C2 C1 C0
SYNC. Mark 3
B8 B7 B6 B5 B4 B3 B2 B1 B0
SYNC. Mark 2
A8 A7 A6 A5 A4 A3 A2 A1 A0
SYNC. Mark 1
LD = 1, iHF = 0, iPTR = "001000000"
SYNC. Mark
4302
Half line
C8 C7 C6 C5 C4 C3 C2 C1 C0
SYNC. Mark 3
B8 B7 B6 B5 B4 B3 B2 B1 B0
SYNC. Mark 2
A8 A7 A6 A5 A4 A3 A2 A1 A0
SYNC. Mark 1
LD = 1, iHF = 1, iPTR = "000000100"
SYNC. Mark

Fig. 43



LD = 1, iHF = 0, iPTR = "001000000"



LD = 1, iHF = 1, iPTR = "000000100"

Fig 44

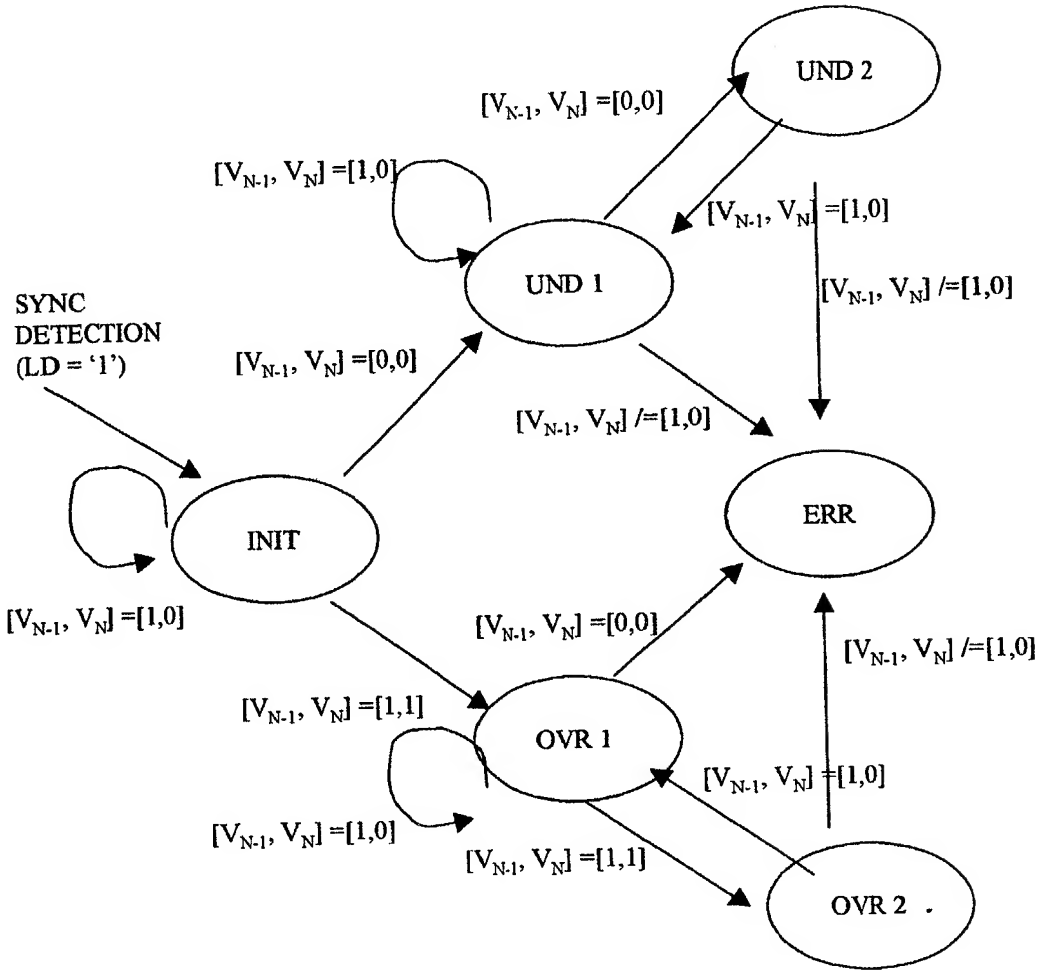

UND 2
$[V_{N-1}, V_N]$ =[0,0]
$[V_{N-1}, V_N]$ =[1,0]
$[V_{N-1}, V_N]$ =[1,0]
UND 1
$[V_{N-1}, V_N]$ /=[1,0]
SYNC
DETECTION
(LD = '1')
$[V_{N-1}, V_N]$ =[0,0]
$[V_{N-1}, V_N]$ /=[1,0]
ERR
INIT
$[V_{N-1}, V_N]$ =[1,0]
$[V_{N-1}, V_N]$ =[0,0]
$[V_{N-1}, V_N]$ /=[1,0]
$[V_{N-1}, V_N]$ =[1,1]
OVR 1
$[V_{N-1}, V_N]$ =[1,0]
$[V_{N-1}, V_N]$ =[1,0]
$[V_{N-1}, V_N]$ =[1,1]
OVR 2

Fig 45

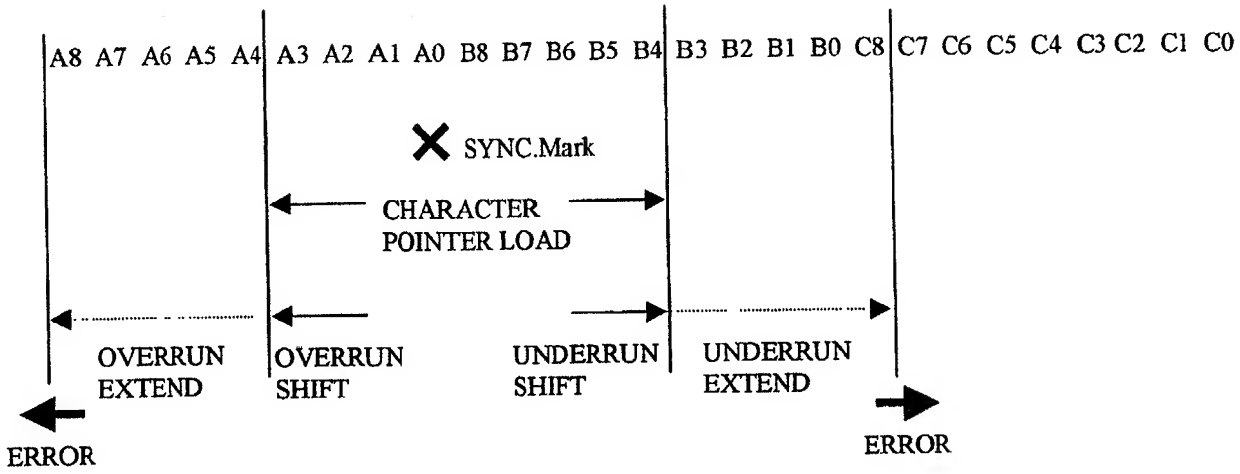

A8 A7 A6 A5 A4 A3 A2 A1 A0 B8 B7 B6 B5 B4 B3 B2 B1 B0 C8 C7 C6 C5 C4 C3 C2 C1 C0
SYNC.Mark
CHARACTER
POINTER LOAD
OVERRUN
EXTEND
OVERRUN
SHIFT
UNDERRUN
SHIFT
UNDERRUN
EXTEND
ERROR
ERROR

Fig 46

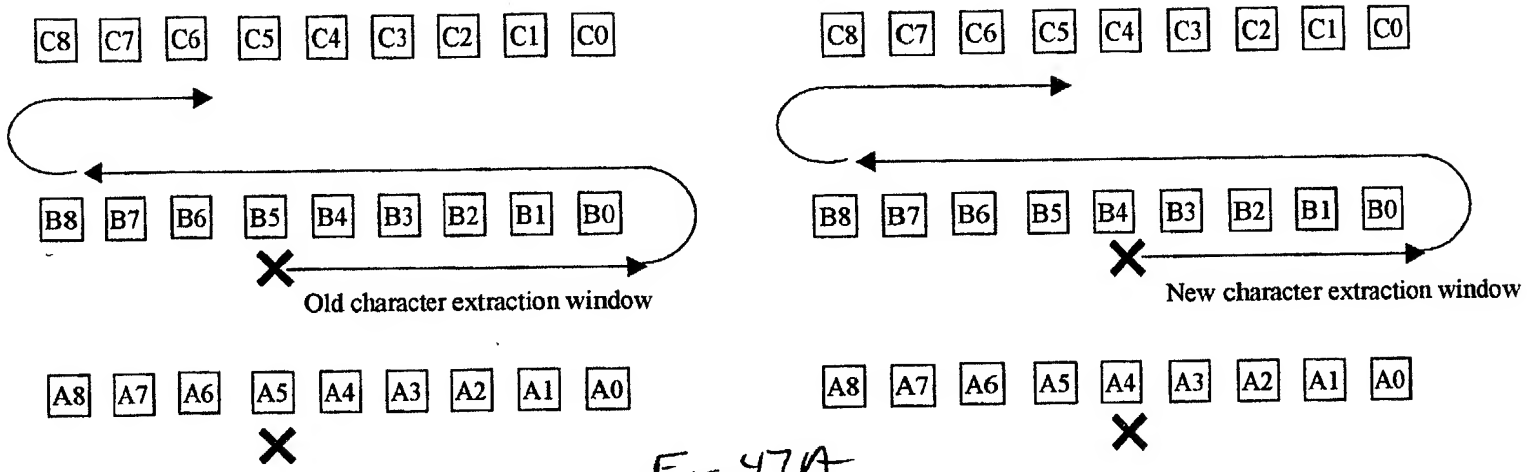


Fig 47A

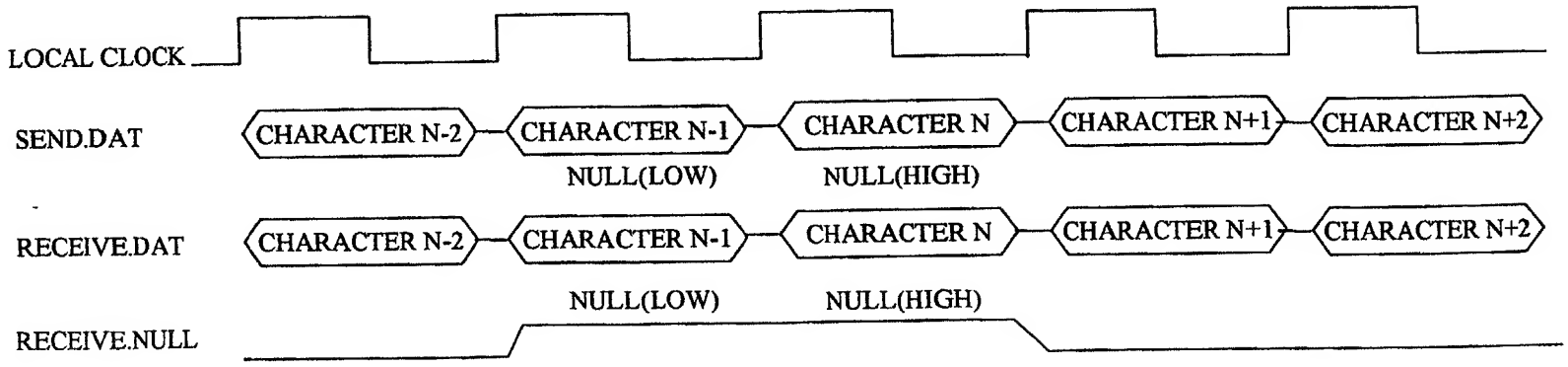


Fig 47B

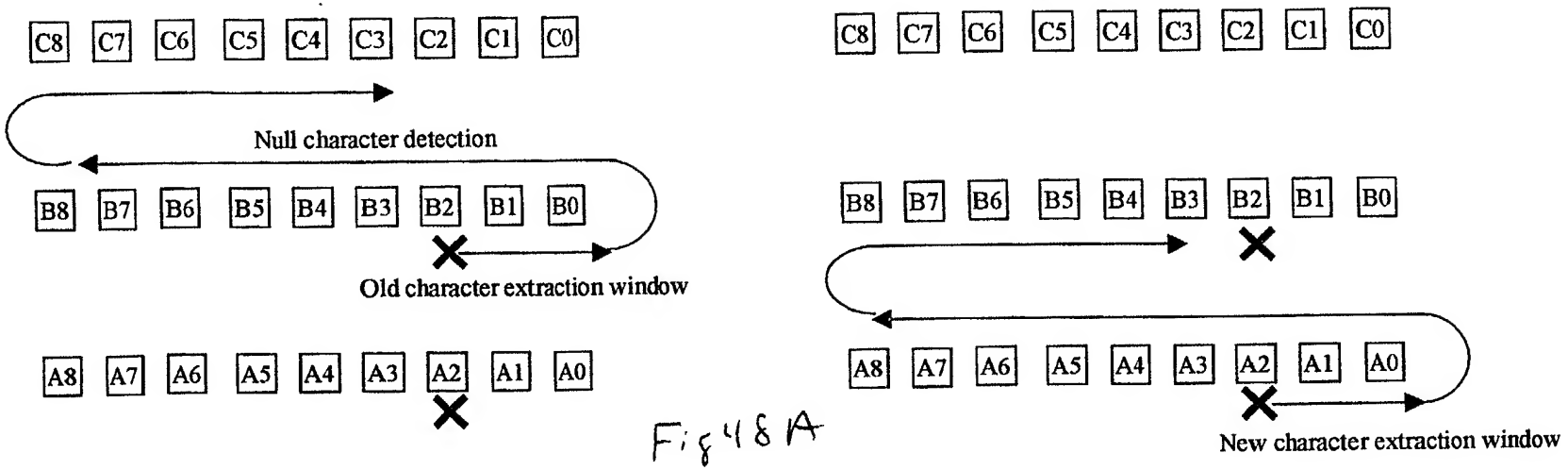


Fig 48A

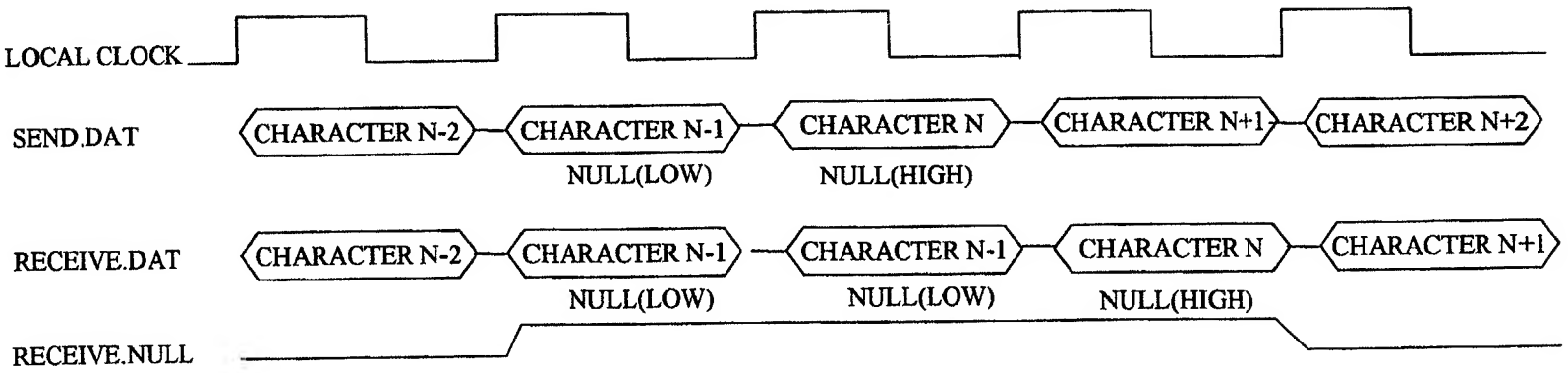


Fig 48B

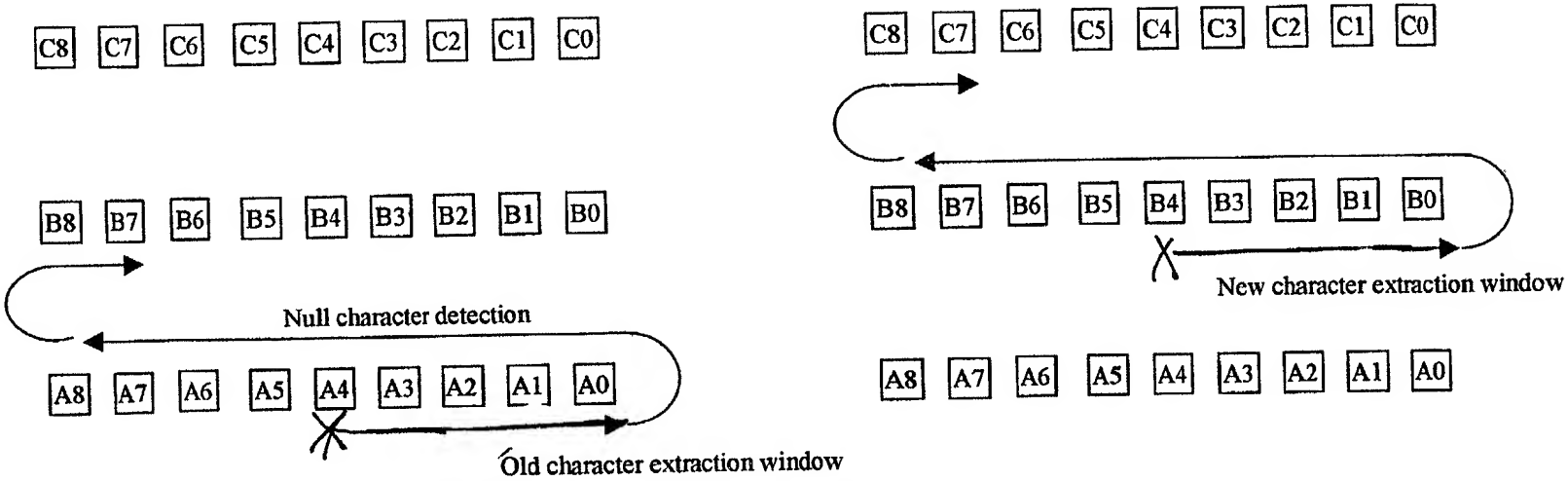


Fig. 49A

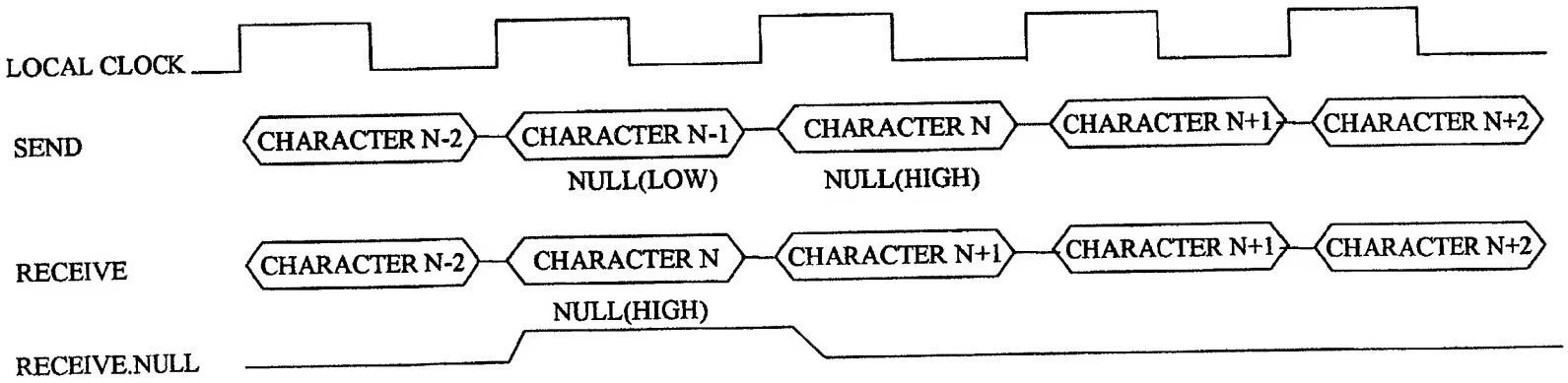


Fig 49B